Erkka Mutanen

Three-dimensional Measurement of a Lifted Load using Machine Vision

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 20.10.2014

Thesis supervisor:

Prof. Ville Kyrki

Thesis advisor:

D.Sc. (Tech.) Sami Terho

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Erkka Mutanen

Title: Three-dimensional Measurement of a Lifted Load using Machine Vision

Date: 20.10.2014 Language: English Number of pages: 10+120

Department of Electrical Engineering and Automation

Professorship: Automation Technology Code: AS-84

Supervisor: Prof. Ville Kyrki

Advisor: D.Sc. (Tech.) Sami Terho

In crane environments, a three-dimensional measurement of the load object is required to develop new automated features, such as route planning, collision detection, and collision avoidance technology. The incentives for developing such features are improved easiness of operation in manually operated crane systems and an increase in goods handling safety.

A 3-D machine vision process was selected to produce a load object measurement using passive stereographic triangulation. A perception platform with two high dynamic range cameras was used to perceive the environment and acquire image data. The acquired image data was processed into 3-D point clouds that represented the surface model of the environment in a time series.

A 3-D bounding volume measurement of the load object was acquired from the surface model using Point Cloud Library(PCL) processing. The measurement software was implemented using C++ programming language and Robot Operating System(ROS). The load object was classified from the image data using a 2-D image tracker, and in some cases a 3-D classification using a proximity-based criterion was used.

Machine vision measurement was analysed using 25 offline datasets from two different industrial environments. In an indoor environment, a measurement from the offline data was achieved with an accuracy of ±15% of the actual load object dimension value. Cylindrical load objects were detected from an outdoor environment using a 2-D image tracker and a RANSAC parameter fitting technique. The load object measurement was successful for cylindrical load objects detected from outdoor offline data. The accuracy of the outdoor application was not analysed.

Keywords: stereo vision, depth triangulation, lifted load, 3-D surface mapping

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Erkka Mutanen

Työn nimi: Kolmiulotteinen mittaus nostetusta taakasta konenäköavusteisesti

Päivämäärä: 20.10.2014 Kieli: Englanti Sivumäärä: 10+120

Sähkötekniikan ja automaation laitos

Professuuri: Automaatiotekniikka Koodi: AS-84

Valvoja: Prof. Ville Kyrki

Ohjaaja: TkT Sami Terho

Nosturin taakan kolmiulotteinen mittaus vaaditaan uusien automaattisten toimintojen mahdollistamiseksi nosturiympäristössä. Uusia toimintoja ovat esimerkiksi reitinsuunnittelu-, törmäystarkastelu-, ja törmäyksenestotoiminnot. Edellä mainittujen toimintojen kehittäminen voi helpottaa käsiohjatun nosturin käyttöä ja mahdollistaa turvallisemman taakan siirtämisen tilassa.

Taakan mittauksen tuottamiseksi valittiin 3-D-konenäkömenetelmä, joka käyttää passiivista stereokolmiointimenetelmää. Työn toteutuksessa käytettiin kahta korkean dynamiikan kameraa, jotka tallensivat kuvadataa nosturiympäristöstä. Kuvadata prosessoitiin 3-D-pistepilviksi, jotka kuvaavat ympäristön pintamallia ajan funktiona.

Taakan kolmiulotteinen rajoittava tilavuusmittaus laskettiin Point Cloud Library -kirjaston(PCL) avulla 3-D-pintamallidatasta. Mittausohjelmisto toteutettiin C++ -ohjelmointikielellä ja Robot Operating Systemillä(ROS). Taakan valinta kuvadatasta tehtiin 2-D-kuvaseurannalla, ja joissain tapauksissa taakan valinta suoritettiin 3-D-datasta yksinkertaisella etäisyyskriteerillä.

Konenäkömittausta analysoitiin 25:llä aineistolla, jotka oli teollisuusympäristöistä tallennettu. Sisätiloissa olevan teollisuusympäristön aineistoilla tuotettiin taakan tilavuusmittaus, jonka tarkkuus oli tilastollisesti $\pm 15\%$ etäisyydellä halutusta tuloksesta. Sylinterimäisiä kappaleita mitattiin ulkotiloissa kaapatusta aineistosta 2-D-kuvaseurantaa ja parametrisovitusmenetelmää (RANSAC) käyttäen. Taakan tilavuusmittaus toimi myös ulkotiloissa tallennetuilla aineistoilla, mutta ulkotilamittausten tarkkuutta ei analysoitu.

Avainsanat: stereonäkö, syvyyskolmiointi, nostettu taakka, 3-D-pintakartoitus

Preface

The research done in this thesis was a part of the FAMOUS project of the Energy and Life Cycle Cost Efficient Machines(EFFIMA) research programme, managed by the Finnish Metals and Engineering Competence Cluster(FIMECC). The project was funded by the Finnish Funding Agency for Technology and Innovation(TEKES) together with research institutes and third party companies. Their support is gratefully acknowledged - thank you!

I joined the research team of the EFFIMA programme in February 2013, and I was super happy to be located in the Department of Automation and Systems Technology (AS) on the Aalto Otaniemi campus. The AS department has been my workplace and school, and all my fellow students who I have studied with were now always just a click away. A big thanks to all my colleagues at the department: you deserve a big bowl of ice cream(if you are not lactose intolerant that is) for giving me your opinions and suggestions and sharing your time in the 2nd floor coffee room.

I would like to thank my thesis supervisor Ville Kyrki, who always inspired me with knowledge and ideas in the field of mobile robotics and machine vision. This applies even in the course work: Ville gives the best lectures that the department students have ever had in the department.

I would especially like to thank my awesome supervisor Sami Terho, mainly and foremost a great researcher, but also a newfound dad, for his untiring guidance to help me realise this thesis. The best advise you gave me was to keep persistently finding solutions - even in the face of science that more often throws more questions at one than solutions.

Other humans I would like to thank: Antti Valli, Juhana Ahtiainen, your software solutions made my day more than once - Toni Liski, the TLD tracker wizard who made the 2-D image tracking possible - Khurram Gulzar, thank you for the ROS sessions - Janne Paanajärvi, for evaluating the feasibility of my machine vision designs and providing the R matrix - Kristen Kiong from National University of Singapore, for proofreading and correcting my English vocabulary - Iris Vainio, for motivating me through hard times and being my Helsinki University agent. A special thanks goes to Annika Salama and Heikki Koivo: without you this book would not have happened! Finally, I would like to thank my family for their awesome support, my other family also known as ASUS for all the good times, and Chicos-club for all the peer pressure and good discussions!

in Espoo, Otaniemi, 26.09.2014

Erkka Mutanen

Contents

Abbreviations and Symbols

Abbreviations

AABB	Axis-Aligned Bounding Box
AI	Artificial Intelligence
API	Application Programming Interface
BM	Block Matching
BVH	Bounding Volume Hierarchies
CoG	Center of Gravity
CPU	Central Processing Unit
CMOS	Complementary Metal Oxide Semiconductor
CSM	Complete Surface Model
CUDA	Compute Unified Device Architecture
DOF	Depth Of Field
DOP	Discrete Oriented Polytope
DP	Dynamic Programming
FOV	Field Of View
FPS	Frames Per Second
GPS	Global Positioning System
GPU	Graphics Processing Unit
HOG	Histogram of Oriented Gradients
HDD	Hard Disk Drive
HDR	High Dynamic Range
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
InF	Infinity
LAN	Local Area Network
LIDAR	Light Detecting And Ranging
MSAC	M-estimator Sample And Consensus
MSER	Maximally Stable Extremal Regions
NaN	Not A Number
OBB	Oriented Bounding Box
OS	Operating System
PAL	Phase Alternate Line
PC	Personal Computer
PCL	Point Cloud Library
PROSAC	Progressive Sample Consensus
RADAR	Radio Detecting And Ranging
RANSAC	Random Sample And Consensus

ROI	Region Of Interest
ROS	Robot Operating System
SAD	Sum of Absolute Differences
SFM	Structure From Motion
SGBM	Semi-Global Block Matching
SIFT	Scale-Invariant Feature Transformation
SURF	Speeded Up Robust Features
TLD	Tracking, Learning, and Detection
ToF	Time-of-Flight
UDP	User Datagram Protocol
VSM	Visible Surface Model
XML	Extensive Markup Language

Symbols

A	camera installation location in datasets A1-A6
b	baseline distance between two camera frame origins
B	camera installation location in datasets B1-B7
C	camera installation location in datasets C1-C7
C	camera matrix
c_l	principal point in left camera image plane
c_r	principal point in right camera image plane
c_x	principal point x-component in metric units
c_y	principal point y-component in metric units
D	camera installation location in datasets D1-D5
$d \vert d(x, y)$	disparity
E	essential matrix
e_i	epipole, $i = 1, 2$
F	fundamental matrix
f	focal length
f_x	focal length x-component
f_y	focal length y-component
h	imaging sensor height in pixels
K	camera calibration matrix
k_c	distortion correction vector
k_{c_i}	distortion correction vector coefficient, $i \in [1\ 6]$
l_1	left epipolar line equation
l_2	right epipolar line equation
O_c	arbitrary camera frame origin
O_L	left camera frame origin
O_R	right camera frame origin

O_w	world frame origin
p	projection point on the camera image plane
P	point cloud(n points)
P_c	coordinate point in camera frame
P_w	coordinate point in world frame
$\tilde{P}$	homogeneous coordinate point
p_l	projection of P in the left camera image plane
p_r	projection of P in the right camera image plane
Q	reprojection matrix(disparity to depth -mapping matrix)
R	rotation matrix
$\tilde{R}$	homogeneous rotation matrix
r_{ij}	rotation matrix component
s_x	photosite width in metric units
s_y	photosite height in metric units
t	translation vector
T_c^{-1}	extrinsic calibration matrix
t_i	translation vector component
u_0	principal point in u-direction in pixel units
v_0	principal point in v-direction in pixel units
w	imaging sensor width in pixels
X_{aabb}	estimated expected value of AABB measurement along the camera frame x-axis
x_p	normalised distortion corrected projection of X-coordinate
x_L	distance of projection p_l from principal point c_l along x-axis
x_R	distance of projection p_r from principal point c_r along x-axis
$x\|y\|z$	projection of 3-D coordinate component
$X\|Y\|Z$	3-D coordinate component
y_p	normalised distortion corrected projection of Y-coordinate
Y_{aabb}	estimated expected value of AABB measurement along the camera frame y-axis
z	depth measurement
α	angle in estimation of expected value geometry
α_c	skew term
β	angle in estimation of expected value geometry
γ	angle between camera frame x-axis and load main axis
$\delta_{x\|y\|z}$	standard deviation of the error along an axis
π_L	left camera image plane
π_R	right camera image plane

Chapter 1

Introduction

In the modern world, products are being manufactured more than ever before, and at the same time the factory workflows should be optimised to guarantee a profitable continuity for a business. These new requirements challenge all business owners, including industries that utilise cranes in the daily business.

Many different types of cranes were designed to help in loading and unloading, moving, and stacking of goods since the industrial revolution. The most popular crane types are electric overhead cranes, ship-to-shore cranes, rotary boom cranes, rotary telescopic boom cranes, chain lifters, double girder overhead cranes, nuclear power station cranes, and person lifters.

The development of modern crane systems and sea cargo have made the lifting and moving of goods more efficient in general, but especially in seaports[1]. Automatic container cranes of the sea cargo process are a good showcase of highly efficient and safe crane systems. In spite of the development of container automation, small cranes often remain manually operated and have a limited capability of independent operation. Small crane operations could become more efficient with semi-automatic functionality in places where currently the crane is operated manually only.

The business of lifting solutions gets more automated as new crane and boom systems are being developed to meet the increasing needs for productivity. Productivity may be measured using selected productivity indicators. For example: energy consumption, service intervals, speed of operations, and safety features are easily quantifiable measures that can add up to a productivity rate of a crane. All of these indicators affect the performance of goods handling, and with the decrease in computing prices a lot of research is focused on finding ways how programmes and sensors could have a positive effect on the productivity of a crane system.

1.1 Interactive Control Of Cranes And Booms

Many different kinds of loads are being lifted and transferred daily in a small crane environment: e.g. beams, engines, large coils, drums of dangerous goods, and pallets of stacked items. The lifted loads are typically unique single items that are valuable and must not be damaged. If a lifted object is in collision, it may get damaged, break other objects, or it may cause dangerous situations to the people who work in the area.

Interactive control can prevent manually operated lifted goods from colliding with the nearby environment. The idea is to introduce duality in control so that the crane user can design high-level operations, and the computer controller optimises the low-level operations e.g. movement and collision avoidance in a semi-autonomous manner. The user executes the complex decision-making at times of lifting and lowering of the load, and the computer controller executes optimal movement of the load within error limits of the user-created design while making sure the trajectory is clear of objects. The computer controller provides feedback of the current state of the operations so that the operator may adjust his or her working style interactively.

A human operator is needed to assist and supervise a computer controller when complex decision-making is at hand[2]. For example, decision-making is needed when the lifted load object must be placed down, and the speed required must not be too high. Other demanding tasks include lifting items off the ground, lifting objects that are stacked, and traversing of narrow spaces(a scenario used in research of autonomous crane controllers). A computer is well-suited to quickly detect changing dynamics and operating a crane or a machine boom at energy-optimised speeds. While fully automatic cranes and machine booms have been built, they only operate in spaces where people are not allowed to work simultaneously. Additionally, existing fully automatic cranes typically move only bulk materials such as sand, steel, or standard-size containers.

A collision avoidance technique can be used with a manual crane system so that the crane controller can avoid other objects automatically in the path of the load object in a dynamic work environment. Ideally, a collision avoidance technique simply prevents collisions by engaging safety measures if needed. The system should be able to detect people who walk near the load and safely slow down the operations if people come too close, because safety in the workplace is of highest priority. The most common safety issues, phases of lifting operations, and general safety engagement using a computer vision system are introduced in more detail in the thesis by Laitasalmi[3].

Without designing the details of the human-computer interaction here, the operator interacts with the computer safety systems so that productive and safe operations can be carried out.

To realise a working collision avoidance system, an understanding of the crane

environment must be formed including all the objects that move nearby the load at any time. In general, all the static and moving parts of the crane environment should continuously provide information about feature location, dimensions, and moving speeds. A stereo camera pair is one possible solution that can solve the load object dimension measurement problem, since it has the capability of measuring objects with sufficient accuracy and providing their location and dimensions from a single stereo image pair. Also, a stereo camera can measure multiple nearby objects simultaneously, which helps in deciding whether any objects will be found on the trajectory of the lifted load.

From a technical point of view, a discrete-time spatial tracker(4-D tracker) needed in the interactive collision avoidance system is difficult to realise. Trackers that would be suitable for use in a collision avoidance system include object state vector management systems[4], and Kalman filter based trackers[5], which are out of scope of this thesis. If the load object is tracked and its three-dimensional measures are known in a sufficient detail, a collision computation with a known environment model can be used to prevent collisions. The adding of a smart collision avoidance feature in a manually operated crane potentially adds value to the user of the crane, which is why the measurement of the load object is worth researching.

1.2 Applications

Typically, industrial applications of machine vision solve problems of ranging, quality control, visual odometry, object detection, and tracking.

Machine vision applications in industrial crane applications may include for example visual servoing[6], real-time trajectory planning[7], wand controller following[8, 9], intelligent fault detection[10], obstacle avoidance[11], load object tracking and control applications[12], remote localisation of the end effector[13], and stereoscopic teleoperation of forestry cranes[14].

The stereo camera platform used in this work also enables visual recording of events. For example, training videos can easily be recorded for new operators, and visual records of incidents may be saved for future reference.

The load object measurement should add functionality to safety applications. A traditional safety application halts the crane process if people are detected in a dangerous work zone[15]. With the help of a load object measurement and a collision control system, a safety mode that does not fully stop the crane can be realised.

1.3 Goals And Objectives

The goal of this thesis is to measure the dimensions of load objects being lifted and moved with the help of a stereo camera based machine vision system. The load

object measurement performance is analysed in two different environments: using indoor lifting equipment in a constructed indoor environment, and using outdoor lifting equipment in an outdoor location.

A primary objective is to implement a load object measurement software that is easy to integrate into other software solutions and that supports networking in an industrial network using PC computers. Primarily, the software measures and reports dimensions of a single load object that is lifted separately in space. Secondarily, the software measures and reports dimensions of a single load object that is stationary on the ground or in a pile of other objects.

An objective regarding hardware is to keep the costs of the load object measurement system significantly lower than the price range of a light detection and ranging(LIDAR) based solution. An objective regarding software is to make the load object measurement client provide measurement data at least multiple measurements per second.

1.4 Research Problem

This thesis studies whether a stereographic machine vision system can be used for three-dimensional measurement of a load object in an industrial environment. The system must be able to measure generic loads that are being lifted using any available lifting equipment. The environments cannot be controlled, and no controllable light sources are being used.

The measurement system must be usable in a constructed indoor environment. The measurement system must be usable also in an outdoor environment where daylight conditions are present. A perception platform that can be freely placed is used to acquire data from the surrounding environment. Stereo cameras are mainly used, thus, image data is being recorded from the environment. The following questions should be studied in this work:

- Is it possible to measure a correct 3-D volume of a load object by only using a single camera viewpoint?
- What issues must be addressed when using a single-viewpoint generated visible surface model as the measurement data?
- Is it possible to measure multiple load objects simultaneously using the stereo cameras?
- Is the accuracy of a stereo camera solution sufficient to be used in a safety application?
- Is there a camera that works in both indoor and outdoor environments?
- What 3-D data representation is the most efficient in a bounding volume computation?

- What are the working steps for generating bounding volume information from point cloud data?
- What camera viewpoints produce the best results for load object measurement?
- Can open-source software be used for stereo camera calibration in this work?
- What block matching parameters should be used to produce the most accurate 3-D reconstruction of the environment?
- What simplifications and assumptions must be made to make the computation possible in the face of mathematical problems and missing data?
- Can cylinder fitting improve the description of the occupied space of a cylindrical load object?
- In what applications is a load object measurement useful in?
- How many load object measurements per second can be achieved with a stereo system?

Chapter 2

3-D Perception And Representation

The problem of measuring a three-dimensional load is that it inherently requires a sensor that can recover a 3-D geometry. This chapter introduces sensor options, acquisition techniques, and representation formats that can be used for 3-D environment acquisition and processing.

2.1 Sensors

Machine vision sensors are used to recover information about the surrounding environment by capturing emitted electromagnetic radiation. Typically, devices such as industrial digital cameras, time-of-flight(ToF) cameras[16] and structured-light cameras are used. Additionally, consumer grade camera products, such as *Kinect* devices, *digital single-lens reflex* cameras, and *webcams* can also be used in a machine vision system. Usually, only industrial grade camera products are used in industrial applications. The consumer grade products, on the other hand, are used in academic research or in environments where continuous operation is not a requirement.

Camera-based vision sensing can be active or passive depending on the technology used. An active sensor emits energy into the environment and measures the energy radiated back to the sensor array. A passive sensor does not emit energy, but it only registers energy radiated by the environment to the sensor array. Typically, the energy that is registered and digitally sampled by a camera is visible light, but sometimes other spectrum of light e.g. ultraviolet spectrum can be used.

Some machine vision sensors output an image while others output a range image or a 3-D point cloud. All machine vision sensors have one thing in common: they record the response of a spectrum of light using a photosensitive imaging array.

Other perception sensors that exist can sense the environment and produce 3-D data from a crane load, such as light detecting and ranging(LIDAR) scanners and radio detecting and ranging(RADAR) sensors. These sensors are not strictly machine vision sensors, because they do not form a 2-D image from the sensing

process. Still, these sensors must be mentioned, because they are a competing perception sensor technology suited for 3-D measurements. For example, a LIDAR scanner can recover an accurate 3-D geometry of the environment, which can be used to verify a camera based solution's measurement accuracy[17]. RADAR technologies are less accurate and more suited for positioning applications, but they are also used for 3-D modeling in air-borne and space-borne geographic modeling and other specialised 3-D modeling applications[18].

A lot of different sensor options are commercially available for machine vision systems. The design of the machine vision system is highly affected by the selection of the sensor and the digital format of the information that is output from the sensor.

2.2 3-D Environment Representations

A *3-D environment model* is a digital data representation that describes the environment acquired by a perception sensor. Efficient modeling of real-world objects has been a major research topic especially in the virtual reality research and in the field of computer graphics rendering. Creation of a digital model of a real-world object automatically is not easy, but many approaches can be found in the literature. The selection of the most suitable digital model depends on the used acquisition hardware. Next, a selection of suitable digital representation format options are presented.

2.2.1 Point Cloud

A *point cloud* is a set of point coordinates that typically describe a 3-D surface geometry of an object(see Figure 2.1). Point clouds are widely used in stereo reconstruction, metrology applications, robotics applications, multiple viewpoint data registration, 3-D data processing, and 3-D visualisation purposes. Point clouds can be acquired using a LIDAR scanner, a stereo camera setup, or other equipment that can output 3-D point data.

Point data in a point cloud can be *organised* or *unorganised*. Organised point cloud data is structured in a 2-D array format, which enables robotic machine vision application use. For example, point cloud colouring using a photograph can only be done if the point cloud structure is organised.

Point cloud geometry can be easily rendered and inspected, but the data is often not usable in a 3-D application without processing. For example, point cloud data may require a conversion to a polygon representation, or a mesh model[19].

Point cloud representations support fast binary data saving, and flexible storing of different primitive data types natively[20]. Some challenges with point clouds include limited suitability for modeling dynamic environments, and high memory

consumption when a high precision equipment is used to generate a large number of point cloud entries[21].

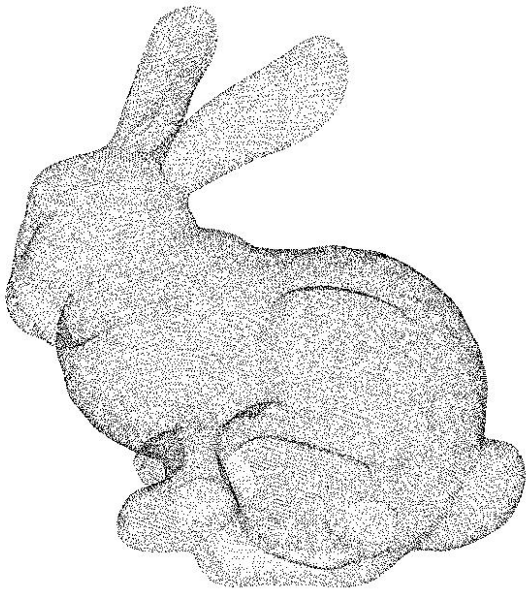

Figure 2.1: A point cloud representation of the *Stanford bunny* dataset[22] from[23].

2.2.2 Range Image

A *range image* is a 3-D representation that has depth measurements in a 2-D grid array(see Figure 2.2). A range image is conceptually equivalent to a standard intensity image, but the intensities are replaced with range values. Range images only support modeling of surfaces seen from the camera viewpoint. Such a model is called a visible surface model(VSM)[24].

The downside of the range image representation is its limited ability to reproduce a complete surface model(CSM). If a CSM that is represented with a point cloud is converted into a range image, all data that is not visible from a selected viewpoint is lost in the conversion.

2.2.3 Volume Pixel Grid

A volume pixel, or *voxel*, describes a location in space. It has a volume, and a location. A grid structure of voxels is called a *voxel grid*, which partitions a 3-D space into voxels. In digital systems, a voxel is a location-based item that has optional services, such as statistical information about the neighbourhood of the voxel. A 3-D voxel grid can be used to represent a 3-D scene. For example, Figure 2.3 shows a voxel grid representation of the Utah teapot.

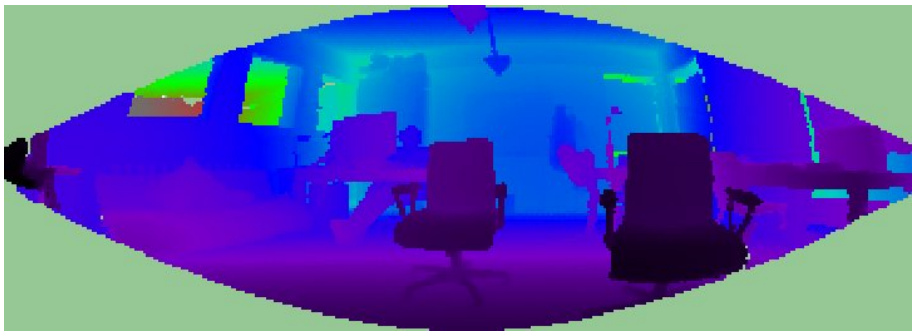

Figure 2.2: A 2-D range image representing an office environment. The depth measurements have been colour coded. Image from Point Cloud Library[25].

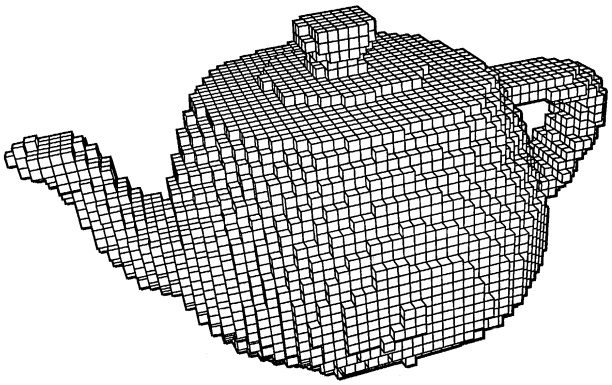

Figure 2.3: A voxel grid representation of the Utah teapot[26].

A downside of the voxel grid representation is its limited grid resolution, which does not approximate curved surfaces well unless a large number of data entries are used. A high-resolution grid, on the downside, would consume a lot of memory in software implementation[27].

Voxels in a voxel grid can either all be the same size or different size. Typically, only single-sized voxels are included in a single scene. A voxel grid resolution value is used to report the detail of the 3-D model. Different resolutions can be used to create high detail or low detail 3-D voxel grid models, analogous to 2-D image pyramids. If a single 3-D scene contains different voxel grid resolutions for different subregions of 3-D space, then more advanced multiple resolution representations can be used, such as an octree representation.

2.2.4 Octree

An *octree* is a tree structure whose nodes always have exactly eight voxel grid child nodes. The octree structure is used to partition 3-D data recursively into smaller

partitions of space around a point region[28]. In an octree, the size of the voxels change depending on the search depth of the tree search in a subregion(see Figure 2.4).

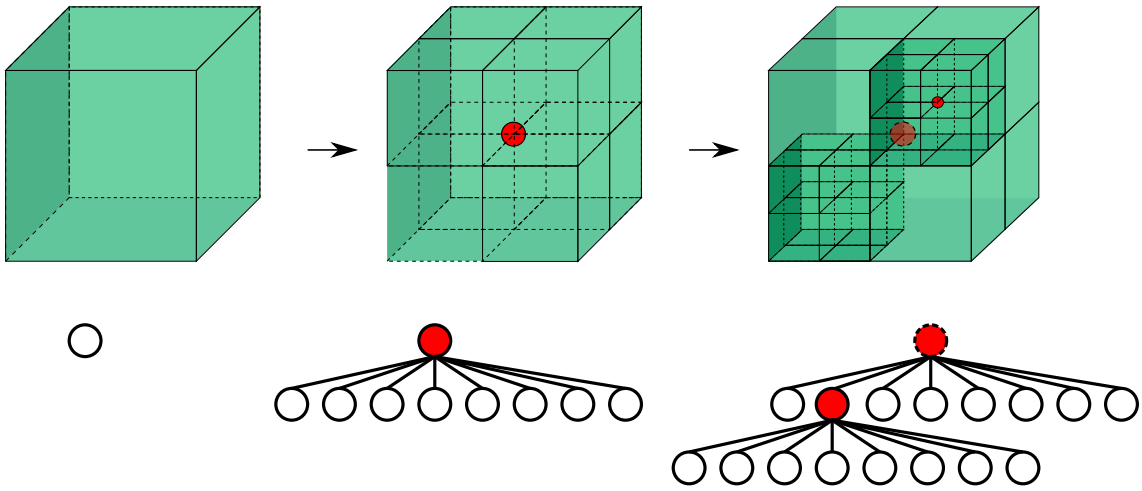

Figure 2.4: An octree structure partitions space recursively around a point region using exactly 8 child nodes. The highest resolution of the octree representation depends on the available number of search depth levels.

Octrees are used in the field of 3-D computer graphics for environment modeling. For example, the *id Tech 6* game engine uses an octree structure for environment modeling[29]. An octree representation of the actual test environment of the load object measurement indoor location is shown in Figure 2.5. The load object is coloured in red, and the green object next to it represents a human standing nearby.

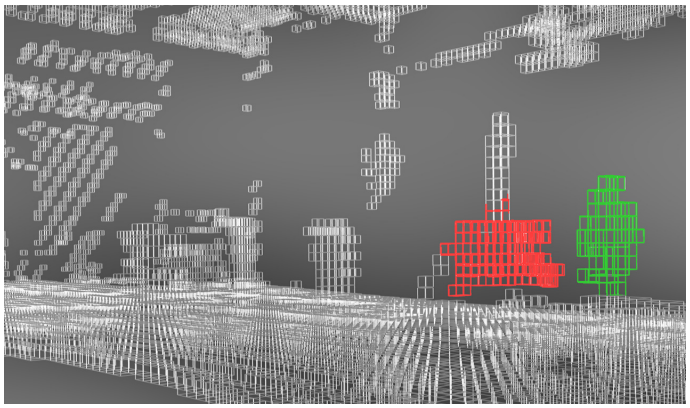

Figure 2.5: An octree representation of an actual indoor lifting equipment environment with a constant voxel cube size. The load object can be seen in red colour, and the green object is a human standing nearby.

Octrees support nearest-neighbour search, efficient collision detection, and set operations(e.g. union, intersection, and difference)[30]. Octree representation is suitable for fast interactive physics simulation or feedback control, because the memory-efficient structure is able to detect virtual object collisions by the milliseconds[31].

The nodes of the octree include states for the regions of space with a *free*, *occupied*, or *unknown* status, similarly to voxel grids. The state property makes the octree structure able to support a detection, classification and tracking of a moving object in a 3-D environment[32].

2.3 Environment Acquisition

Environment acquisition focuses on systems capable of recovering the geometry of surrounding environments in 3-D. Natural environmental factors that are not visible in an image, such as temperature, wind conditions, humidity or other qualities of the environment are not considered in the context of 3-D acquisition, although they have an effect on the perception sensors. Triangulation and structure from motion(SFM) are passive sensing techniques that are introduced as the basic mechanisms that most 3-D acquisition techniques are based on.

2.3.1 Real-world 3-D Modeling

In general, a real-world scene is difficult to model using perception sensors. A large number of images from a single environment is required to compute a virtual environment model that displays a correct geometry of the environment[33]. Images can be recorded using multiple perception sensors, or a single moving sensor.

The quality of the environment model may degrade in bad weather e.g. fog, mist, rain, or snow. Camera systems are especially affected by inhomogeneous degradation in image quality. Different types of degradation in image quality are caused by issues such as lens flare, condensation of water on lens, changes in illumination, shadows, and noise from droplets of water moving on the lens. Weather also affects the surrounding environment: for example, rain droplets change the surface reflectance properties of visible surfaces. If coloured markers are used in a tracking system, the sensitivity to changing natural light and weather conditions must be taken into account[34].

A 3-D environment model can fully model the 3-D surfaces of an environment or it can model only select surfaces. A single viewpoint visible surface model (VSM) is simple to acquire, and the more completeness of modeling is needed the more computation the model requires. A complete surface model(CSM) of a 3-D -environment can be built using multiple VSM depth maps, but usually only to a partial completeness. Multiple depth maps that describe different parts of the environment must be *registered* using the iterative closest point algorithm(ICP)[35] in order to compose

a single CSM model using an *integration* method. Multiple-view registration is a classic technique that can estimate more complete geometric surfaces of objects in comparison to a single camera viewpoint generated VSM. Multiple-view registration is out of the scope of this thesis.

2.3.2 Triangulation

Triangulation finds distance to a point by measuring two angles from known points that are located at two ends of a fixed baseline. Triangulation is based on the theory that the known camera viewpoints and the unknown point form a triangle whose one side and two angles are known. Before the global positioning system(GPS) was engineered, the mapping of land areas (calculating distances and directions of landmark features) was performed using *triangulation networks* with the triangulation technique.

The term triangulation is originally used in trigonometry calculus, but in machine vision systems it refers to the indirect process of measuring image feature depths using two sensors. A triangulation system that uses two cameras is a passive stereo triangulation system. One that uses a single camera and a light pattern projector is an active triangulation system. In this thesis only passive triangulation techniques are used, which is why only the two-camera simple stereo geometry will be introduced in Section 3.5.

2.3.3 Structure From Motion

Structure from motion(SFM) is a machine vision technique that uses only a single moving camera that reconstructs a 3-D geometry of a scene using a sequence of captured images. Many similar techniques exist which are applications of the SFM technique e.g. structure from silhouette[36].

Structure from motion is similar to biological visual systems that easily infer 3-D structure from motion with little a priori knowledge of the world. For example, if a person closes one eye, it is possible to understand the 3-D geometry of the surrounding environment only by moving in it, and the structure from motion mechanisms in the brain compute the correct geometry without binocular vision.

An SFM system is able to reconstruct static scenes with a moving camera, but reconstructing a dynamic scene is more challenging. In the case of reconstructing dynamic environments with a single camera, some additional problems must be addressed, such as what image regions correspond to moving objects and which do not, and how does the illumination change spatially and temporally. Due to the limitations of modeling spaces that contain moving objects, SFM is not a suitable technique in modeling of a dynamic lifting equipment environment.

2.4 Camera Optics

The machine vision cameras include optical lens and shutter systems that control the amount of light entering the photosensitive CMOS sensor array in the camera. The most important parameters that must be set in the optical system include focal length, field of view(FOV), aperture size and depth of field(DOF), and camera shutter speed. Other parameters that may affect the performance of the camera system are the selection of lens materials, and how light enters through the interface of different lens materials. The camera parameters are often set by the selected camera hardware. Still, some parameters can be controlled, for example, focal length can often be adjusted independently in a camera lens system.

The aperture of the camera lens system controls the DOF perceived in a digital image. The DOF attribute affects the range of depth that appears in sharp detail in the output image. If the DOF is large, then only a small depth range is in sharp detail, and objects in the other depth ranges will appear blurred in the output image. If the DOF is small, then all objects seen in the image will be in sharp focus.

For the design of a stereo camera depth measurement system, a small DOF, or *extended DOF*, is used to fully enable a deep depth range of a scene in sharp detail. The load object will stay in sharp detail before and after a lift operation, even though the load object is found at different depths in the camera frustum at each time. A trade-off between feature sharpness and the amount of noise generated in the image occurs when all depth ranges are desired to be shown in sharp detail. Since aperture sizes are limited by diffraction, images captured with a small DOF may suffer from high variance noise effects and low light conditions. The aperture of the camera is described by a unitless f-number (also *focal ratio* or *relative aperture*) that is presented as f over a number. Increase in the f-number decreases the aperture diameter, which means that the extended DOF limit is at the diffraction limit of the aperture. Higher f-numbers have a larger depth of field and the camera lens approaches the asymptotic limit of the pinhole camera model[37, p.1141], which will be discussed in Section 3.2.

Chapter 3

Stereoscopic Machine Vision

The current chapter details how 3-D data can be acquired with two calibrated cameras and what techniques, theories, and models can be used to achieve a successful 3-D data output. The chapter starts with introductions to stereoscopic perception, and a pinhole camera model that includes intrinsic and extrinsic camera calibration corrections. Next, stereo triangulation with epipolar geometry is introduced in more detail, and stereo calibration error sources are identified. Moreover, a stereo correspondence problem is introduced together with its solutions, including a disparity mapping technique. Finally, a reprojection of the disparity map back into a 3-D point cloud is shown.

3.1 Stereoscopic Perception And 3-D Reconstruction

Stereoscopic perception is a natural phenomenon first discovered in human and animal vision systems. Stereoscopy in human vision is based on stereopsis effect, which means perceiving depth in a scene using binocular vision with two monocular eyes. The same depth perception can be programmed to a computer, but instead of eyes, cameras are used. Instead of the brain, a PC is used to recover and process depth information from a pair of images. A calibrated two-camera system that is capable of extracting depth information from objects is called a *stereo vision system* or a *3-D vision system*. 3-D vision systems in engineering applications are typically used to solve 3-D scene reconstruction problems, understanding of object properties, or minimisation problems[38].

The theory of understanding 3-D objects from their 2-D image projections and processing them into information was first described by Marr in 1982 in his book *Vision*[39]. Marr presented a theory, also known as *Marr's theory*, in which it is stated that the models of objects and their surfaces should be understood before a 3-D reconstruction is done. As opposed to Marr's theory, the 3-D reconstruction described

in this chapter processes data independently of its description. Thus, it is not known what the data represents during the 3-D reconstruction. Marr's theory suggests that the representation of objects will improve when a pixel-based viewer-centered intensity image description is transformed into a feature-based object-centered description. The transformation to a feature-based object description is done using segmentation and classification principles described in Chapter 5.

A 3-D geometry can be recovered without any feature model knowledge, but the classification must be done later to identify the load object correctly. The 3-D scene reconstruction starts with camera calibration according to a pinhole camera model, which is introduced in the following section.

3.2 The Pinhole Camera Model

The *pinhole camera model* maps 3-D objects onto a 2-D image plane using a pinhole camera geometry. A three-dimensional point in real space $\mathbf{R}^3$ is mapped into a two-dimensional projection in $\mathbf{R}^2$ according to $\mathbb{R}^3 \mapsto \mathbb{R}^2$, also known as a *perspective projection*. The depth component z of a 3-D point is lost in the transformation, which is natural for any projective camera models. The transform is irreversible, which means that the original 3-D scene cannot be recovered from a single 2-D projection image. The pinhole camera model is introduced because it is the model implemented in Chapter 4, and it is the simplest approximation that can be used to describe the mathematical model of an actual camera image formation[38, 40].

3.2.1 Pinhole Camera Geometry

The pinhole camera is a simple box that has an ideal pinhole aperture on one of its sides. The pinhole camera is a camera without a lens, and it forms an upside down image of a target on the image plane π(see Figure 3.1) on the opposite side of the aperture. The positive z-axis of the camera frame (origin at O_c) ideally coincides with the camera optical axis I_o so that the optical axis is perpendicular to the image plane and pointing towards the imaged scene. The dashed line in Figure 3.1 represents the camera optical axis I_o.

In Figure 3.1, according to the similar triangles that have the red light ray as their hypothenuse along the z-direction, the projection of the Y-coordinate point of P on the image plane becomes

$$\frac{Y}{-Z} = \frac{-y}{f}$$

that leads to a solution of the projected y-coordinate in the camera frame

$$y = f\frac{Y}{Z} \tag{3.1}$$

and similarly for x-coordinate

$$x = f\frac{X}{Z} \tag{3.2}$$

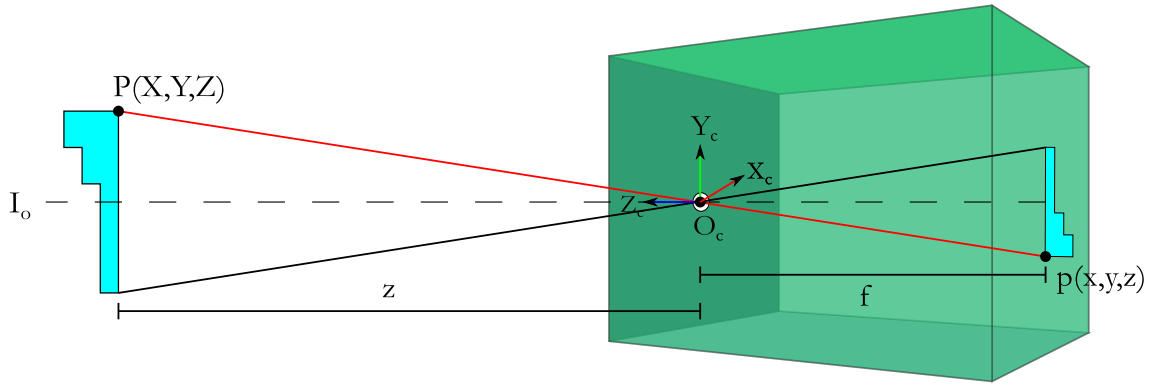

Figure 3.1: A pinhole camera projection. The result (3.1) can be formulated using the pinhole aperture as the origin of the camera frame O_c.

(3.1) and (3.2) are the fundamental equations that describe the geometric projection of 3-D points onto a 2-D image performed by a pinhole camera[41]. A general perspective projection model according to (3.1) and (3.2) can be written as

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \tag{3.3}$$

where X, Y, and Z are the values of a 3-D point P in the camera frame C.

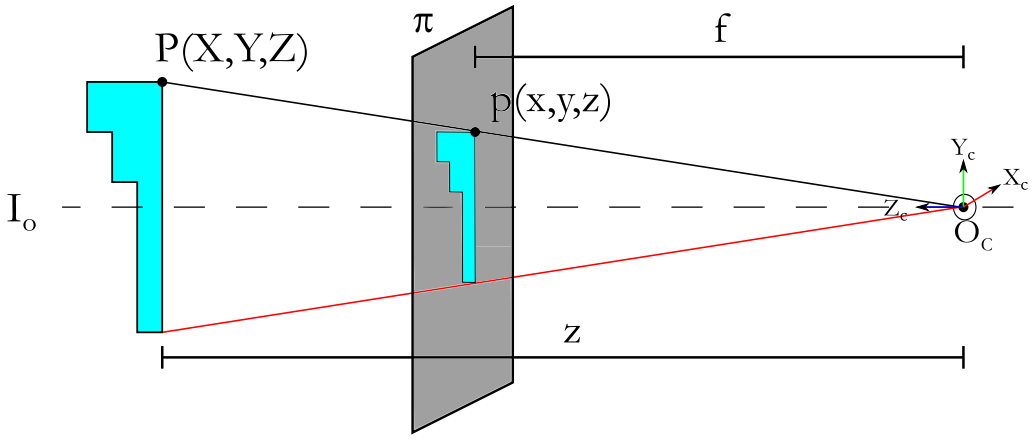

Figure 3.2: A central projection model whose image formation is equivalent to a pinhole camera image formation in Figure 3.1.

Figure 3.2 shows a *central projection model* that projects points similarly to the projection in Figure 3.1. In a central projection model, the resulting virtual

projection image is not inverted, which removes the minus sign from coordinate transformations and leads to a camera matrix C whose diagonal elements are positive(see camera matrix (3.8)). In a pinhole camera, the real image is formed behind the pinhole at plane $z = -f$, whereas in the central projection model, the focal point is used as the center of projection and the virtual image is formed in front of the pinhole camera at plane $z = +f$. A central projective transformation can be represented in a matrix form if *homogeneous coordinates* are used. In homogeneous form, any 3-D point P can be written as:

$$\tilde{P} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

where X, Y, and Z are the 3-D point coordinates of P. Augmenting (3.3) with a projective element $(x, y, 1)^T$ leads to a homogeneous perspective projection

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \\ \frac{Z}{f} \end{bmatrix} \tag{3.5}$$

If the positive scaling value $\frac{f}{Z}$ in (3.5) is ignored, then an equality up to a scaling can be defined as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} X \\ Y \\ \frac{Z}{f} \end{bmatrix} \tag{3.6}$$

which can be expressed in matrix form as

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.7}$$

In (3.7), the camera matrix appears as the 3×4 matrix more commonly shown in form

$$C = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.8}$$

Since stereo calibration later requires the model for both intrinsic and extrinsic parameters separately, a general projection model including both intrinsic and extrinsic matrices can be written as

$$\tilde{p} = KT_c^{-1}\tilde{P} \tag{3.9}$$

where K is the *camera calibration matrix* containing intrinsic parameters, and T_c^{-1} is the extrinsic calibration matrix containing rotation and translation of the camera from the world frame origin. Matrices K and T_c^{-1} can be combined into a single 3×4 camera matrix C that performs translation, rotation, scaling, and a perspective projection[42]. (3.9) can be written out as

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.10}$$

where u and v are the coordinates of projection points in *OpenCV* notation[40] and s represents the scaling term omitted in (3.6).

When multiple coordinate points are perspective projected into an image frame, the transformations can be computed with the same camera matrix C used for a single point transform. If the point cloud P contains n points, a matrix can be written so that it contains the set of points as n column vectors in format

$$\tilde{P} = \begin{bmatrix} X_0 & X_1 & X_2 & \dots & X_n \\ Y_0 & Y_1 & X_2 & \dots & Y_n \\ Z_0 & Z_1 & X_2 & \dots & Z_n \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} \tag{3.11}$$

Multiplying point cloud P with the camera matrix C will result in a 3-D point cloud transformed into an image projection. The inverse process of projecting 2-D coordinate points is called *reprojection*, which is introduced at the end of this chapter. Next, it is shown how the millimeter units are transformed into pixel units.

3.2.2 Extension To Pixel Units

After formulating the perspective projection coordinates in millimeters(or any other chosen metric unit that can be converted to and from pixel units) the projected point $\tilde{p}$ must be related to the pixel units that the digital imaging sensor grid contains. The image sensor contains light sensitive photosites that are arranged in an array whose width w and height h determine the resolution of the camera. It is assumed that in a modern digital camera, the pixels are rectangular with no skew, which means that the angle between the width and height component of a photosite is strictly 90°. The pixel coordinates are usually in a non-negative u-v-coordinate frame whose origin is located at the center of the top left corner pixel of the imaging array. This coordinate frame is called the *image frame* and it coincides with the planar image projection plane π of Figure 3.1.

In an ideal situation, the imaging array is planar, and its photosensitive sites are rectangular. The camera frame coordinates can be calculated from pixel values according to

$$x = -(w_i - u_0)s_x \tag{3.12}$$

and

$$y = -(h_i - v_0)s_y \tag{3.13}$$

where x and y are coordinate values in the camera frame C, w_i and h_i are the width and height of an image point in pixel coordinate values, u_0 and v_0 are the principal point in pixel values, and s_x and s_y are the effective size of the pixel in millimeters[41]. Additionally, it is assumed that $s_x = s_y$.

The principal point (u_0, v_0) of the image sensor is the center of the sensor array where the principal ray casts through the pinhole aperture perpendicularly to the image plane. If the principal point is shifted due to lens alignment error (in a real camera where a lens is often found instead of a pinhole), the error can be corrected by changing the metric value for principal point parameters u_0 and v_0 in the camera calibration matrix

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

where f_x and f_y are the orthogonal focal length components, and u_0 and v_0 are the principal points for the camera. The K-matrix can be written with a pixel skew correction value in component $K_{1,2}$, but with modern precise semiconductor technologies the u and v axes should be precisely orthogonal, which is why in (3.14) the skew term is left zero here[42]. The coefficients of the camera calibration matrix K in (3.14) contain the intrinsic camera parameters. If the coefficients of K are known, it is possible to extract metric values from the image plane[38]. If a more complex camera model is used, the matrix K may have larger dimensions, but the intrinsic parameters are still the coefficients of K.

3.3 Intrinsic Camera Calibration

Intrinsic camera calibration is the process of estimating the unknown coefficients of the camera calibration matrix K in (3.14). Ideally, the calibration technique should estimate the camera parameters in an unbiased manner. Currently used calibration methods assume unbiased estimation, which is not true for most techniques, but the results can still be good[43].

The intrinsic calibration parameters for the stereo camera setup can be computed using for example Jean-Yves Bouguet's *Camera Calibration Toolbox for MATLAB*[44]. The camera calibration toolbox uses an internal camera model described by Heikkilä and Silven[45]. The camera calibration toolbox includes a simple polynomial lens distortion model that extends the pinhole camera model. Next, the lens distortion model used by the camera calibration toolbox is introduced.

3.3.1 Lens Distortion Model

When a real camera lens system with imperfections is used for image capturing, lens distortions must be taken into account before a 3-D reconstruction is done using the image data. If a high quality camera is used, lens distortions may not be large, and only a simple correction model is needed.

The most typical geometric lens distortions are *radial distortions* that arise as a result of the shape of the lens, and *tangential distortions* that arise from the physical assembly of the camera[46]. Radial distortions are rotational symmetric, and cause radial displacement of the scene features as a function of distance from the image optical center. The effect of radial distortion is a *barrel distortion*, a *pincushion distortion*, or a linear combination of both.

To remove the effects of lens distortions, a 5×1 distortion correction vector k_c that contains tangential and radial distortion coefficients is formulated. The radial distortion coefficients correct a barrel or pincushion distortion depending on the type of distortion that is present. The tangential distortion coefficients calibrate decentering of the image and other defects in the compound lens structure.

Other lens distortions, such as optical aberrations (chromatic aberration, spherical aberration, coma) and astigmatism arise from different wavelength light refractions in a lens system[47], and they are not corrected for in this model.

A 3-D geometry reconstruction accuracy is mostly hindered by geometric distortions. A lens distortion model that corrects for radial and tangential distortions will be presented. The *Camera Calibration Toolbox for MATLAB* adds the lens distortion model to the perspective projection according to

$$\begin{cases} x_p = f_x\Big(x_d + \alpha_c y_d\Big) + c_x & (3.15) \\ y_p = f_y y_d + c_y & (3.16) \end{cases}$$

where coordinate components x_d and y_d are the distortion corrected projections of X and Y. The distortion correction is based on the calibration procedure described in Heikkilä's paper[45]. The distortion corrected point coordinates x_d and y_d are normalised in the toolbox, and they incorporate the distortion correction vector k_c according to

$$\begin{cases} x_d = \left(1 + k_{c_1} r^2 + k_{c_2} r^4 + k_{c_5} r^6\right) x + dx & (3.17) \\ y_d = \left(1 + k_{c_1} r^2 + k_{c_2} r^4 + k_{c_5} r^6\right) y + dy & (3.18) \end{cases}$$

where r is short for $r^2 \equiv x^2 + y^2$, and dx and dy are the tangential distortion vectors

$$\begin{cases} dx = 2k_{c_3} xy + k_{c_4}\left(r^2 + 2x^2\right) & (3.19) \\ dy = k_{c_3}\left(r^2 + 2y^2\right) + 2k_{c_4} xy & (3.20) \end{cases}$$

and all parameters k_{c_i} are coefficients of the 5×1 distortion coefficient vector k_c.

Bouguet's camera calibration toolbox provides lens distortion models up to a 6th order polynomial, but according to Bouguet[48], it is not recommended to use the 6th order distortion model for standard FOV cameras, therefore, a 4th order polynomial was used in the calibration step of the stereo pair in the *Himmeli* platform(see Section 4.3).

The intrinsic parameters that were found using the camera calibration toolbox are presented in Table 3.1 for the left camera and in Table 3.2 for the right camera. The results are used to initialise the calibration values for stereo calibration in Chapter 4.5. Figure 3.3 shows the resulting estimate from the calibration for an integrated radial and tangential distortion vector field in the right camera of the stereo pair. A more in-depth analysis of the modeling of radial and tangential distortions may be found in the original publication by Brown[49].

3.4 Extrinsic Camera Calibration

The *extrinsic camera calibration* describes the position and orientation, or *pose*, of the camera in an arbitrary world coordinate frame. The transformation between the world coordinate system and the camera coordinate system can be estimated with a translation and a rotation as was seen in (3.10).

In stereo vision applications, extrinsic camera calibration can additionally be used to describe the relative transformation from the left camera frame origin (O_L) to the right camera frame origin (O_R). For example, Figure 3.4 shows the positions and orientations of all camera frames that were used to compute the transformation from O_L to O_R. If an arbitrary world point P_w is found in a world coordinate frame whose origin O_w is explicitly defined in some world location, the translation and rotation into a camera coordinate system can be formulated as

$$P_c = R(P_w - t) \tag{3.21}$$

Table 3.1: The results for the left GigE μEye camera calibration parameters from *MATLAB Camera Calibration Toolbox* (rounded with 7 digits).

Focal length	f_x	470.5395354
	f_y	470.0528248
Principal point	u_0	361.4851212
	v_0	297.1134158
Skew coefficient	α	0.0000000
Distortion coefficients (5×1 vector)	k_c	[-0.0073703;
		0.0248006;
		0.0050783;
		0.0011502;
		0.0000000]
Focal length uncertainty	δf_x	3.1313055
	δf_y	3.0924522
Principal point uncertainty	δu_0	1.8698244
	δv_0	1.9314148
Skew coefficient uncertainty	$\delta\alpha$	0.0000000
Distortion coefficients uncertainty	δk_c	[0.0036557;
		0.0032499;
		0.0009925;
		0.0009502;
		0.0000000]

Table 3.2: The results for the right GigE μEye camera calibration parameters from *MATLAB Camera Calibration Toolbox* (rounded with 7 digits).

Focal length	f_x	483.9712800
	f_y	483.2710360
Principal point	u_0	379.5935499
	v_0	308.8688683
Skew coefficient	α	0.0000000
Distortion coefficients (5×1 vector)	k_c	[-0.0002056;
		0.0181529;
		0.0047928;
		0.0000514;
		0.0000000]
Focal length uncertainty	δf_x	2.9385693
	δf_y	2.8655623
Principal point uncertainty	δu_0	2.0605689
	δv_0	2.0783195
Skew coefficient uncertainty	$\delta\alpha$	0.0000000
Distortion coefficients uncertainty	δk_c	[0.0036847;
		0.0026363;
		0.0010737;
		0.0010460;
		0.0000000]

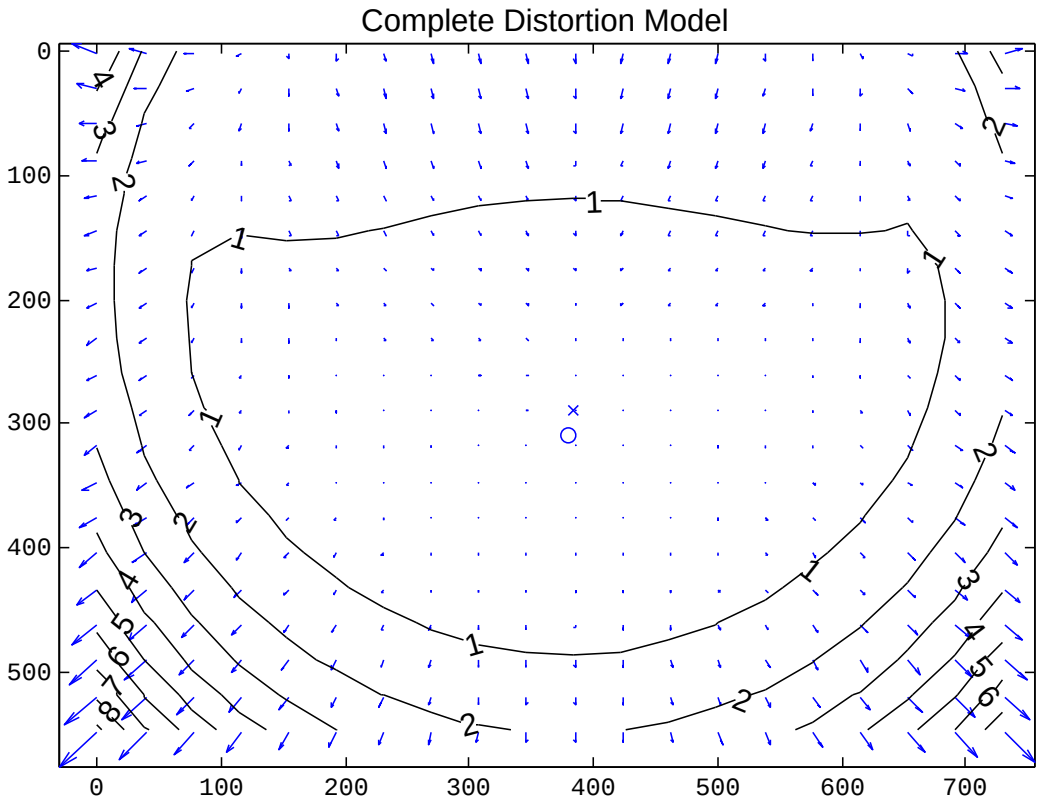

Figure 3.3: Right camera lens distortion model displayed using a vector field. The vector length shows how many pixels an image feature is being displaced in the image caused by lens distortions. The blue cross indicates the image center, and the blue ring indicates the location of the principal point.

where point P_w is transformed into a camera frame using rotation R and translation t. The homogeneous coordinate frame transformation is formulated as

$$\tilde{P}_c = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = R\tilde{P}_w \tag{3.22}$$

where R is a 4×4 matrix. Rotations can be applied in stages instead of a single R matrix using the *Euler angle convention*. In the Euler angle convention the rotations are applied to a rotating coordinate frame, and as such the order of the rotations changes the outcome of the final orientation. The order of applying Euler angle rotations does not matter if the matrices are commutative, but commutativity is not a common property in transformation matrices. The combinations of Euler angle rotations are presented in detail in *Introduction to Robotics* by Craig[50, p.374].

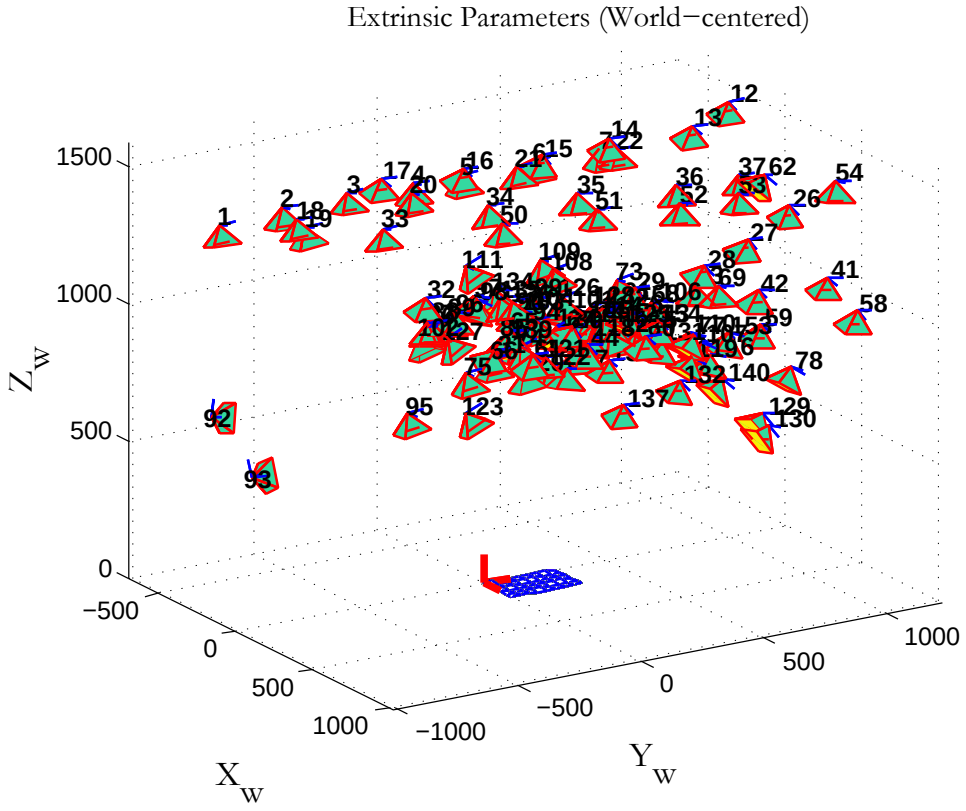

Figure 3.4: Left camera locations seen in a calibration grid centered world frame. The calibration was successful when the calibration grid was at a distance of 0.5-1.5 meters from the camera.

3.4.1 Coordinate Frames

The scene that the camera sensors are viewing is understood as the *world frame*, which uses a standard right-hand 3-D -coordinate convention. The world coordinate frame follows the framework used in a report by Terho published in 2010[51]. The world frame origin O_w is fixed to a predefined location, for example, to a corner of a workspace. In this work, the origin of the world frame is used coinciding with the *camera frame* origin O_L since the load object measurement computation does not require absolute coordinate values.

Relative coordinates are sufficient as long as metricity of the scene geometry is preserved and absolute coordinate values are not needed for positioning purposes. The extrinsic calibration between the left and the right camera should be accurately estimated to guarantee metricity to a correct scale. Third frame, the *image frame*, is needed to convert between metric camera frame locations and the pixel array values

of the image.

The *image frame* is the 2-D projection plane where the 3-D points of the imaged scene are transformed with perspective transformation and pixelation process. The plane of the image frame is fully coinciding with the sensor array. The origin of the image frame is in the upper left corner of the sensor array in order to have only non-negative integer values in pixel mapping.

3.5 Stereo Triangulation

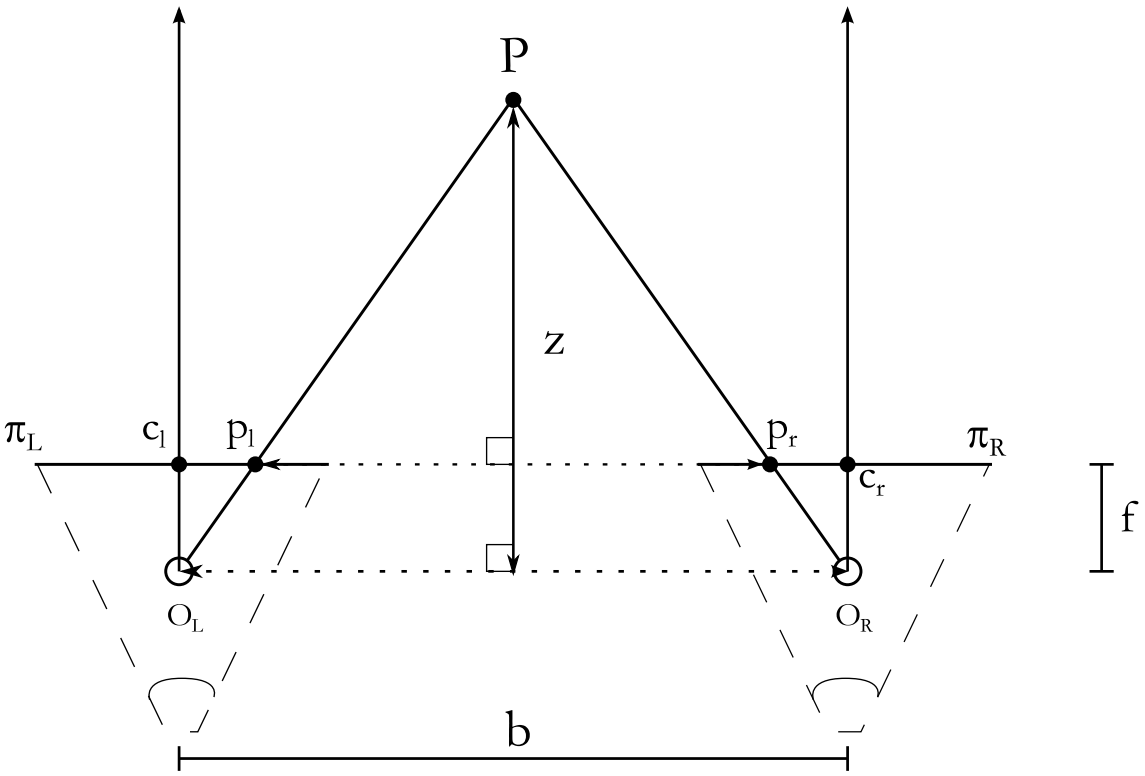

Figure 3.5: Simple stereo geometry shown in a central projective model. Distance z to an arbitrary point P may be solved using similar triangle geometry. Image adapted from [41].

Triangulation is highly dependent on the outcome of the correspondence problem solution(detailed in Section 3.9). Assuming that the correspondence problem is solved, the distance to a point P may be calculated using a two-camera geometry according to Figure 3.5. π_L and π_R are the image planes for the left and right cameras accordingly, and p_r and p_l are the projections of the point P on the adjacent image planes.

Figure 3.5 shows two similar triangles on the left camera geometry whose hypothenuses are vectors (O_L, p_l) and (O_L, P). The z-coordinate of point P is related

according to the similar triangles so that

$$\frac{\frac{1}{2}b - x}{Z} = \frac{x_L}{f} \tag{3.23}$$

where

$$x_L = p_l - c_l,$$

b is the baseline between camera focal points, and x_L is the distance of the projection p_l from principal projection point c_l. Similarly, the right camera geometry will give a relation

$$\frac{\frac{1}{2}b + x}{Z} = \frac{x_R}{f} \tag{3.24}$$

$$x_R = c_r - p_l$$

By definition, disparity is the shift of the image feature in the x-coordinate direction, which can be written[38]

$$d = x_R - x_L. \tag{3.25}$$

Now combining (3.23) and (3.24) and eliminating x gives

$$z(x_R - x_L) = bf \tag{3.26}$$

and taking (3.25) into account gives

$$z = \frac{bf}{d} \tag{3.27}$$

where f is the focal length of the parallel stereo camera pair, b is the baseline of the stereo pair, and d is the measured disparity shift between stereo image features. (3.27) suggests that when disparity is zero, the depth z will be at infinity. According to this model, disparity cannot become negative, which is a simplification of a *canonical camera configuration*. A stereo camera setup where the camera optical axes are in parallel is called a canonical configuration. If a *converging camera setup* is used, then the disparity is exactly zero at the optical axes convergence point, which is found at a finite distance value along the positive z-axis. If the distance from the convergence point increases towards positive infinity of the z-axis, the disparity value increases just like in a canonical configuration. If the distance from the convergence point increases towards negative infinity of the z-axis, the disparity value will become negative. Thus, results shown in (3.23) through (3.27) only hold for a (parallel) canonical camera configuration.

3.6 Epipolar Geometry

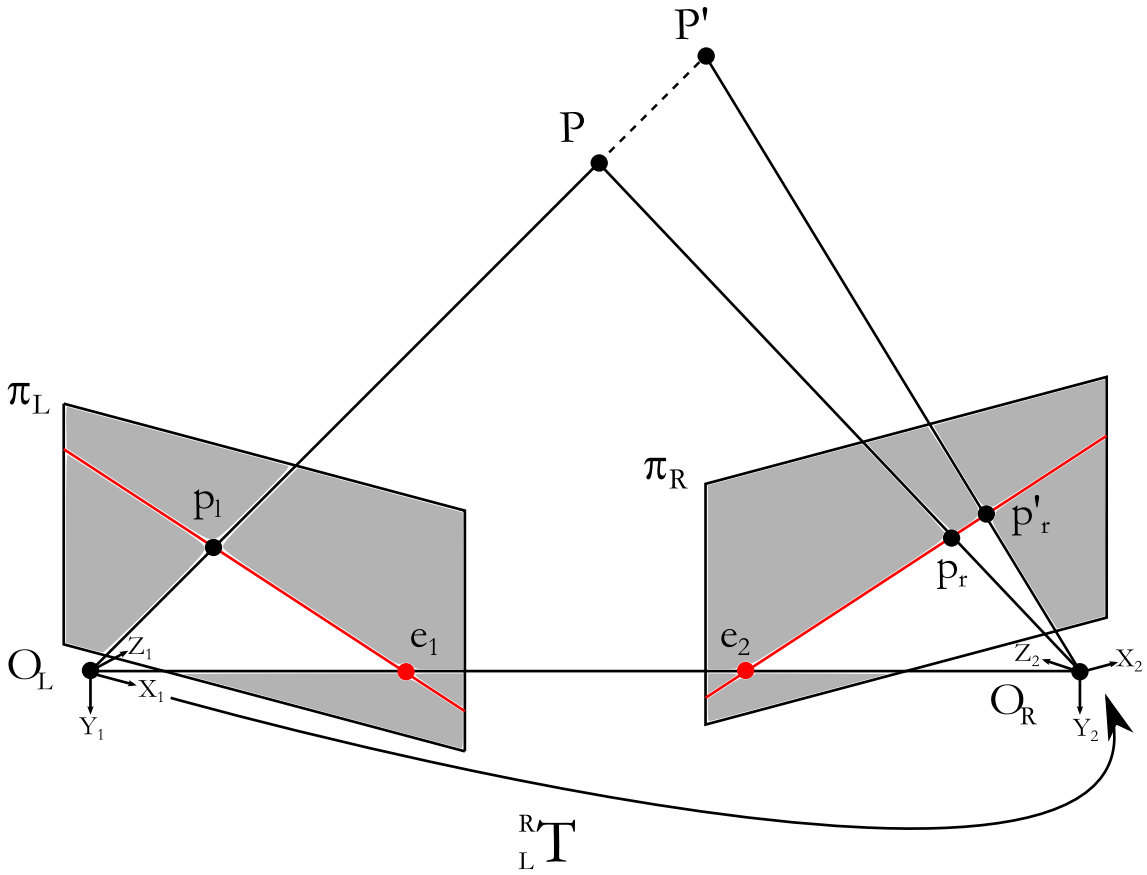

Figure 3.6: Epipolar geometry(central projection). The image planes are shown non-parallel so that the epipoles e_1 and e_2 stay inside the Figure for demonstration. In a parallel configuration, the epipoles are ideally located at infinity. Transformation T_c from the left camera origin to the right camera origin is shown. Adapted from [42].

Epipolar geometry is a two-view camera geometry where the corresponding image point locations are restricted by an epipolar plane. In Figure 3.6, the epipolar plane is the plane spanned by both camera focal points(camera frame origins O_L and O_R) and an arbitrary world point P. The epipolar plane PO_LO_R intersects the image planes π_L and π_R at epipolar lines(see red lines in Figure 3.6). A single point P is projected as points p_l and p_r in the two different image planes. These projected points are called *conjugate points* that are found for all points P visible in both camera images. The most important feature of the epipolar geometry is that the conjugates of a point P in the camera frame must lie on the corresponding epipolar lines, a fact which reduces the correspondence search space from 2-D to 1-D in block matching processing.

Figure 3.6 shows a converging (toed-in) stereo setup where the camera baseline and the epipolar lines in the image planes intersect inside the imaging area. In reality, the epipoles are found at a finite distance outside the image projection plane, but for convenience, Figure 3.6 displays the epipoles. An epipole is not usually found near the image plane except when large FOV cameras are used, and an epipole inside the image plane causes a rectified image of infinite extent, which is undesirable[52]. In Figure 3.6 the epipoles e_1 and e_2(red dots) are located along the epipolar line. An epipole is always found on the stereo camera pair baseline, thus, by definition the epipolar line and the camera baseline intersect at an epipole.

In the canonical configuration the epipoles move to infinity, and the effect known as *disparity* can be seen on the two resulting images. Disparity is defined as the shifting property of image features taken with a canonical stereo pair. For true disparity effect the feature must only shift along a single coordinate axis of a camera frame. Since the cameras project the world in a perspective projection on the image plane, the disparity effect follows the rules of perspective projection. For example, objects further away from the camera optical center shift less in pixels than objects in the near field of the camera. Any stereo configuration with non-parallel optical axes can be transformed into a canonical one by *image rectification* as long as the camera images have sufficient overlap.

A precise relationship between a conjugate pair p_l and p_r is expressed using a 3 $\times$ 3 *fundamental matrix* F according to

$$\tilde{p_l}^T F \tilde{p_r} = 0 \tag{3.28}$$

where $\tilde{p_l}$ and $\tilde{p_r}$ are the conjugate points expressed in homogeneous coordinates. The line equations for the epipolar lines can be solved using F so that epipolar line l_2 is a function of point $\tilde{p_l}$:

$$\tilde{l_2} = F\tilde{p_l} \tag{3.29}$$

Similarly, the epipolar line l_1 can be solved according to

$$\tilde{l_1} = F^T\tilde{p_r} \tag{3.30}$$

Many techniques can be used to recover the fundamental matrix F. If the camera intrinsic parameters are known, then the camera calibration matrix K may be used to recover F. If no calibration is available, then F can be recovered using equation (3.28) with a calibration technique[53]. Methods that can be used to estimate the coefficients of the fundamental matrix are the normalised 8-point algorithm, the 7-point algorithm, and random sample consensus based parameter estimators, such as RANSAC, MSAC, and PROSAC[54].

3.6.1 Image Rectification

Planar image rectification is the process of virtual optical axis parallelisation for projective planar camera geometries. Rectification transforms the image planes so that the same features seen in both stereo images are perceived on the same epipolar lines. Besides planar image rectification, cylindrical and polar rectification can also be used for non-planar camera projective geometries[55].

Rectification of an image pair requires an estimated fundamental matrix F or a set of corresponding feature point coordinates found from both images that can be used to estimate F. Using the fundamental matrix F, the epipolar line equations can be computed according to (3.29) and (3.30). The original non-rectified images are warped to epipolar-aligned destination images, where the pixels that do not receive a mapped value are interpolated using a bilinear interpolation method[56]. The effects of the image synthesis in the destination image depends on the used interpolation method, and more information on different interpolation methods can be found in Grevera's study on interpolation techniques[57]. Some interpolation methods available for use in the image rectification are bilinear interpolation, inter-area interpolation, a bicubic interpolation over a 4×4 kernel, and a Lanczos interpolation over an 8×8 kernel[56].

A stereo camera pair cannot be aligned fully parallel due to limits in mechanical alignment of the lens systems and the construction of the stereo pair. Software rectification compensates for such errors, thus, an accurate positioning of the cameras is often not necessary - rectification is always needed no matter how accurately the cameras have been mechanically aligned. If one of the horizontal stereo pair cameras shifts vertically, then some *vertical disparity* is introduced in the image features, which usually has only minor effect on the image rectification. If one of the horizontal stereo pair cameras tilts around the camera frame x-axis such that the principal rays' nearest interdistance becomes large, the rectification procedure corrects the tilt, but the usable image area, or the region of interest(ROI), in the rectified image will become smaller. ROI is the largest possible rectangular effective image area that can be cropped from the rectified image.

Also, the more the *keystone* correction is needed in case of converged camera setup, the more image synthesis will affect the quality of the output rectified image. Keystone is the distortion effect that is seen when a planar object at an angle is projected on a surface.

A successful image rectification can be seen in Figure 3.7 on the right side where the resulting image will have a smaller ROI than the original(black borders on the resulting image show the original size of the image), and the features will be found on epipolar lines in the matching left and right camera pair images. The larger the ROI is in the resulting rectified image, the more there is image detail left, and the image usable area loss is minimised. For comparison, a rectification result after an

Figure 3.7: An unsuccessful camera calibration result generates large deformations in the rectified image on the left. A successul camera calibration is shown on the right.

unsuccessful stereo calibration is shown on the left side of Figure 3.7. Unsuccessful camera calibration attempts were caused by datasets that did not contain enough data of valid chessboard images. For example, the datasets that contained bright specular lights on the calibration pattern generated unsuccessful calibration results as shown in Figure 3.7.

3.7 Stereo Calibration

A *stereo calibration* computes an extrinsic calibration that relates two camera viewpoints of a stereo camera pair to each other as was described in Section 3.4. The calibration output is a rotation matrix R and a translation matrix T according to (3.22), and the combined transformation can be seen in Figure 3.6. Stereo calibration techniques are based on fundamental matrix estimation techniques that use epipolar geometry and constraints introduced in Section 3.6.

If the stereo camera pair is used to produce metric data output, then both left and right camera intrinsic calibration parameters are required as an input to the stereo calibration. The same algorithms that are used to estimate the fundamental matrix in Section 3.6 can be used to calculate the stereo calibration including the 7-point algorithm, the 8-point algorithm, and random sample consensus based parameter estimators methods.

3.8 Stereo Errors And Accuracy

A stereo 3-D reconstruction system accuracy is limited, which means that any geometry output will include additional noise and errors in the discrete data point entries. If only uncertainty in the measured disparity is considered, an uncertainty model can be formulated using a partial derivative of depth z. Starting with (3.27) a partial derivative of z over d gives

$$\frac{\partial z}{\partial d} = -\frac{fb}{d^2} \tag{3.31}$$

A depth uncertainty Δz can be approximated as

$$\Delta z \simeq \frac{\partial z}{\partial d}\Delta d \tag{3.32}$$

and combining equations 3.31 and 3.32 yields

$$\Delta z \simeq -\frac{fb\Delta d}{d^2} \tag{3.33}$$

where Δd is the resolution of the disparity difference detection in the block matching, down to the limit of the subpixel accuracy, for example, 0.1 pixels. It can be seen from (3.33) that the accuracy of the z-measurement is linearly proportional to the disparity measurement error. Additional trade-offs exist between other factors. For example, an increase in the baseline distance between the cameras will increase the measurement error in (3.33). Accordingly, the disparity will increase, and its term in the denominator is quadratic, which ultimately decreases the error.

In standard stereo processing the object measurement error increases with items that are located far away. If a single baseline and resolution is used, the error grows quadratically with increasing depth measurement[58]. Increasing the baseline width will in general decrease the error although the change will affect the size of the overlapping image area and the disparity space. A stereo setup that has a baseline much higher than the human vision system baseline width is called a *hyperstereo* setup[59]. A hyperstereo setup makes feature matching more difficult than with a smaller stereo baseline setup because the changes between the stereo images are not small and incremental. Block matching is difficult especially for moving objects, such as humans or swaying trees.

The baseline width can be chosen arbitrarily as long as the two images include enough overlapping image area to enable a correspondence problem solution. The stereo resolution depends on the overlapping image area, which is why a large baseline is not usually favourable.

If the cameras are taken far apart, the near-field will lose image coverage, thus, the selection of the baseline width is a trade-off. For example, if the baseline width is large, all nearby objects can be reconstructed only using a converging camera setup.

A comprehensive study of the effects of sub-pixel interpolation, vertical camera misalignment, matching kernel size, focus, maximum disparity, and stereo baseline width on the stereo correspondence can be found in a Jet Propulsion Laboratory technical report from 2005[60].

3.8.1 Occlusion

Occlusion is a major error source in 3-D reconstruction processing. Surfaces that lie inside the camera 3-D frustum, but are not visible in the projected 2-D image, are called occluded objects.

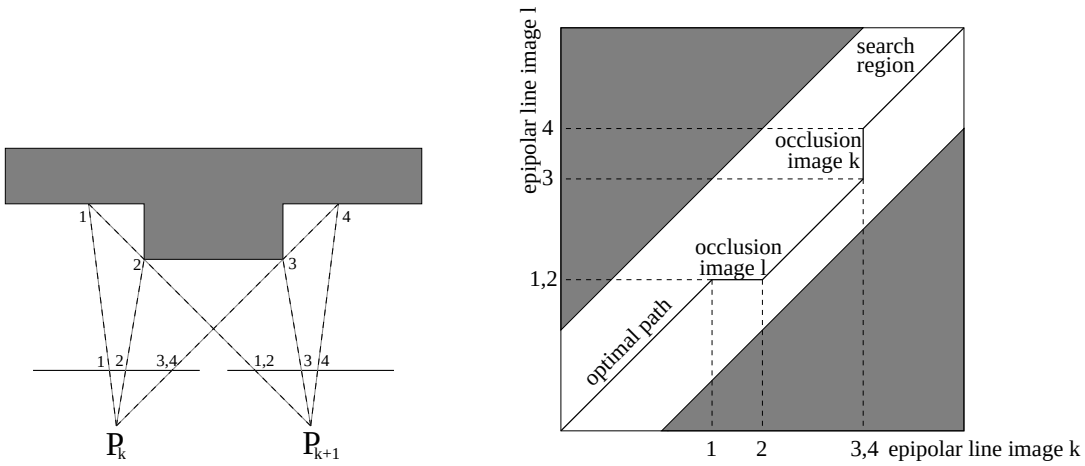

Figure 3.8: a) An occlusion front is seen in a scene of planar features that have depth differences. b) An optimal correspondence correlation path is found within a search region along epipolar lines. Image from *Visual 3-D Modeling From Images* by Pollefeys[61].

Occlusion means partial or full loss of line of sight of an object in an image. An object may be occluded in only the left or the right stereo image, or be partially occluded in both. In stereo matching, an object may be partially occluded by another object, or the object surfaces that curve out of view can self-occlude. In both cases, the other stereo image has visible object surface that cannot be matched in the other image. The correspondence search should not return a correspondence solution for an occluded part of a surface. Typically, occluded regions generate small occlusion patches called *speckles* in the output disparity map in machine vision systems.

Figure 3.8a shows an example of an occluded part of a wall, and Figure 3.8b depicts a correspondence search space along epipolar lines with an ideal correspondence correlation path for the rectified occluded scene. It can be seen from Figure

3.8b that when an occlusion is found in the scene, the correct corresponding image point is found further away from the middle of the search region. Furthermore, the correspondence search space is restricted along the epipolar lines with a certain disparity difference limit, or bandwidth, which is seen as the white stripe in Figure 3.8b[61].

Most solutions do not explicitly detect occluded regions, thus, errors may be found in the disparity output of block matching if *bidirectional matching* is not enforced. Bidirectional matching searches corresponding surface points from both images. Occluded regions may be identified if certain surface areas are detected as *missing surfaces*. Also, an *occluding boundary* can be found on the perimeter of curving objects, for example, a cylinder surface will curve until the other camera view becomes occluded. Occlusion induces errors in the disparity map estimation, which can be improved if occlusion detection is used in the block matching phase. One solution of how to detect occlusion is by using energy functions that optimise the correspondence search path, for example Kolmogorov et al. describe such a technique that uses a *graph cut* technique[62, 63].

3.9 Stereo Correspondence Problem

The *stereo correspondence problem* is a two-camera problem that answers which parts of images are projections of the same object surface[41]. Humans are good at solving the correspondence problem - the same object can be easily found in an image pair, which is a result that a computer system will not solve easily.

Stereo correspondence can be solved using *correlation-based methods*, or *feature-based methods*. Feature-based methods only provide sparse matching algorithms that match distinctive features. Correlation-based methods can be extended to dense matching that run for all pixels[51].

The correlation-based methods rely on the assumption that corresponding features have same intensity values in different images(and the same intensity pattern in a local neighbourhood), and they can be matched using an image kernel. *Block matching* algorithms are an example of correlation-based methods, also known as area-based stereo methods. The block matching algorithm is used in the software implementation in Chapter 5, and is detailed more in Section 3.9.2.

Feature-based methods use sparse feature points found using feature detectors, such as the *Harris corner detector*, or a SURF descriptor. A feature-based correspondence can be found using a *PMF algorithm*, which outputs correspondence matches that can attain sub-pixel accuracy[38].

Constraints can be used to reduce the size of the correspondence search space, for example, the epipolar constraint, uniqueness constraint, disparity smoothness constraint, disparity limit constraint, ordering constraint, and mutual correspon-

dence constraint are used in the *OpenCV* implementation of the prototype software block matching. A short taxonomy of the available correspondence constraints can be found in the book *Image Processing, Analysis, And Machine Vision* by Sonka et al in Chapter 9[38].

3.9.1 Feature Description

In a 2-D image, a *feature* is a 2-D projective response of a 3-D object that is formed on the image plane according to the camera model described in Section 3.2. Since different cameras have different viewpoints of the same 3-D object, the feature response of the object is not the same for each viewpoint.

The feature response may be simplified and extracted using image segmentation techniques such as histogram or colour-based thresholding, k-means clustering, water-filling algorithm, region growing algorithm, or some other segmentation method and matched using feature descriptors.

A *feature descriptor* measures the shape of a segmented image region using invariant moments of the image, or other constant shape descriptors. A good feature descriptor response is invariant to changes in scale, orientation, and viewpoint, and it should tolerate changes in the camera system used for imaging, such as white balance, exposure, and illumination changes such as specular reflections and shadows[42].

Image feature detection is a large field on its own, which is why in this work only a few general feature descriptors are introduced that can be used to track features and solve the correspondence problem. Some full-fledged feature descriptor techniques include scale-invariant feature transform(SIFT)[64], speeded up robust features detector(SURF)[65], and histogram of oriented gradients(HOG)[66]. Also blob detectors, such as the maximally stable extremal regions(MSER)[67] could be used to solve a general correspondence problem[68]. The Harris corner detector[69] and Shi and Tomasi corner detector[70] are useful for tracking the corners of a lifted object as part of the measurement software. Moreover, the features returned by a corner detector could be used as control points in the point cloud verification process introduced in section 6.2.

3.9.2 Block Matching

Block matching algorithms are area-based stereo methods that solve the stereo correspondence problem using the epipolar constraint introduced in Section 3.6. A *block matching algorithm*(BM) that uses a mean square error metric, and a *semi-global block matching algorithm*(SGBM) that uses a modified Hirschmuller algorithm[71] are present in the load object measurement software that was designed in this work.

The block matching algorithm assumes that the cameras are in a canonical con-

figuration with parallel optical axes, a condition which is enforced using image rectification by software. The algorithm blocks the left and the right image into kernel size blocks, which are minimally 1 pixel, and usually 3×3 pixels wide kernel windows. Each block is assigned a mean intensity value and being compared with sliding window responses from the second image. When a selected error metric, e.g. squared mean error is being minimised, a correct match is found, and the disparity value is assigned accordingly. Block matching algorithm generates a sparse correspondence match matrix. If *scalable coarse matching* methods are used, then it is possible to extend the correspondence search to a dense matching scheme.

In this work, a sliding kernel of a single pixel, or larger single row kernel window is a requirement since the correspondence search was reduced to a 1-D scan line search by the epipolar constraint.

In general, a small kernel window is desirable to avoid unnecessary smoothing in the block matching, but optimal window size depends on the region intensity variations and texture richness. For example, in low contrast image regions a small window cannot guarantee a unique match due to little intensity variation. The trade-offs in a window size selection can be solved with iterative window size methods. Still, increase in computation overhead, and other problems such as occlusion boundary and matching on specularity edges, will remain unsolved[72].

3.10 Disparity Mapping

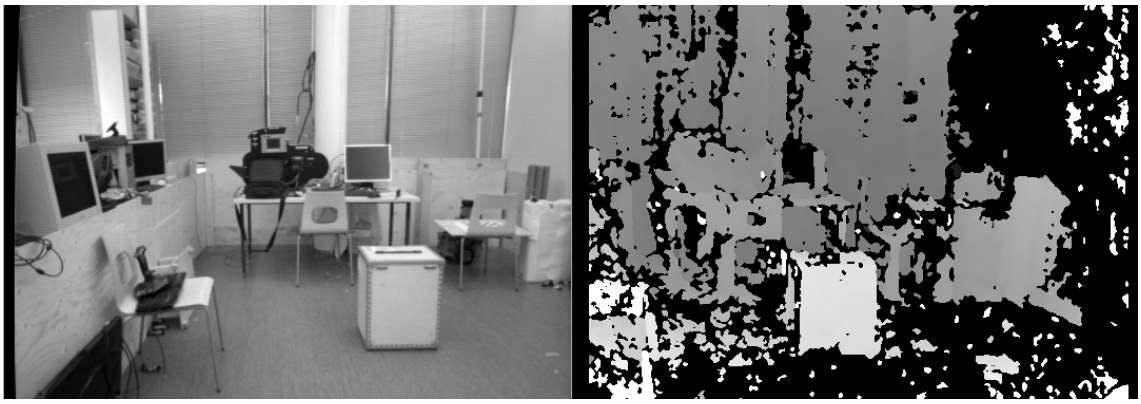

Figure 3.9: An office environment and its disparity map.

The 3-D reconstruction problem can be formulated as a problem of extracting the best possible estimate of a disparity map $d(x, y)$. A *disparity map* is a single channel image that contains the disparity values computed in the block matching processing, and it can be associated with one of the stereo camera images - often the left camera image being used. Figure 3.9 shows a disparity map that was generated

by the used hardware from an office environment. The expected ideal disparity map that the viewed scene should generate is called a *ground truth* disparity. In reality the correspondence matching can only recover an approximation of the ground truth.

The quality of the disparity map depends on the selected block matching algorithm, and on the selected block matching state parameters. The block matching state include 10 parameters that are discussed more in Section 6.3. The full parameter descriptions and value ranges can be seen in Table 6.1.

Using the pinhole camera model and the inverse of its camera projection matrix C (3.8) the disparity map values can be reprojected back to 3-D coordinate points. The reprojection matrix Q is a 4×4 transformation matrix that maps disparity to depth[73]. In *OpenCV*, Q-matrix can be estimated using the function *stereoRectify()*. Finally, the 3-D point cloud is generated by reprojecting the disparity map values according to equation

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = Q \begin{bmatrix} x \\ y \\ d(x,y) \\ 1 \end{bmatrix} \tag{3.34}$$

which solves the 3-D reconstruction problem. *OpenCV* provides readily available function *reprojectImageTo3d()* to generate the point cloud accordingly.

3.10.1 Disparity Mapping Simplifications And Assumptions

A 3-D reconstruction using a calibrated stereo triangulation system is not possible without some simplifications and assumptions. In general, the assumptions shown here hold for any stereo camera setups that work on a triangulation principle and use disparity mapping.

First, it is assumed that a single pixel must correspond to a single surface point in the world in order to construct a linear mapping function from the left stereo camera image to the right camera image. This restriction means that no transparent, or see-through materials can be accurately measured with the used technique. The load objects being measured are assumed as Lambertian surfaces that do not change in appearance when viewed from another angle. A *Lambertian object* is opaque material that supports ideal diffusion of light and reflects light energy in all directions. Additionally, occluded surface pixels should be discarded in the block matching phase to comply with the single pixel correspondence assumption.

Secondly, disparity values are assumed as being continuous, which means that the local disparity neighbourhoods have a smoothness restriction. Typically, high frequency edges in disparity maps will be smoothed out, causing sharp 3-D features in the environment being reprojected as curved features in the 3-D reconstruction of the original environment.

Finally, it is assumed that discontinuities only occur at object boundaries. A more detailed report for disparity map assumptions and requirements can be found in a report by Zitnick and Kanade[72].

Chapter 4

Software Design And Hardware

The software design and hardware chapter begins with a brief introduction of the Robot Operating System(ROS) and proceeds to show how the overall software architecture was designed. ROS is introduced, because the software was built upon the ROS platform. Next, the perception platform and its cameras are detailed, and the locations where the platform was installed in the test sessions are listed. Finally, the *OpenCV* stereo camera calibration procedure used in this work is detailed.

4.1 Robot Operating System

Robot Operating System(ROS) is an open-source operating system framework targeted specifically for robotics application use. ROS is structured with a hierarchy of nodes, packages, and community-supported repositories that provide the user with a large code base of readily available robotic applications. ROS system can also utilise a heterogeneous computation network, which is administered using a master-slave network in a local area network(LAN). ROS was originally developed by the Stanford AI laboratory in 2007, and currently it is supported by the open-source community.

ROS was used as a data transport medium in the test software due to its simple yet powerful messaging capabilities that use a *topics* and *subscribers* model. Data can be disseminated via topics in the ROS network, and any configured computer can access and subscribe the information as needed. Additionally, ROS provided readily available debugging tools for networking issues, and an optional server-provider model for future use. The final load object measurement software can easily be integrated to many environments using a ROS package distribution.

ROS was the first choice for networked data communication, because it was already used in the other parts of the same research project. Other options were also considered: a new implementation of the user-datagram communication protocol(UDP), a real-time operating system(such as FreeRTOS), or an implementation

with no networking capabilities at all. ROS was selected for use in the software development effort because the other options were difficult to realise in a PC network in an industrial LAN setting.

4.2 Software Architecture

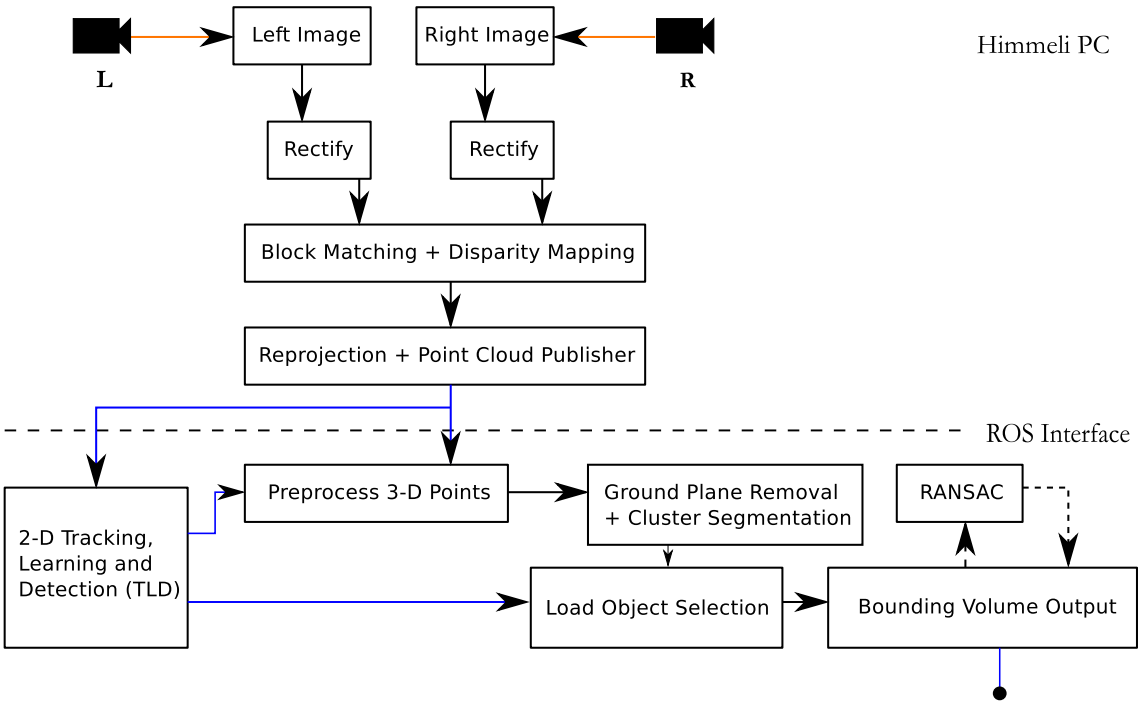

Figure 4.1: Software architecture for the load object measurement workflow using Robot Operating System. The blue lines depict data transmissions via ROS messages, and the orange lines depict UDP data transmissions.

A stand-alone software client for the load object measurement was designed and implemented that processes point cloud data into load object bounding volume information. Additionally, a point cloud data publisher was programmed that generates point cloud data from the *Himmeli* perception platform stereo camera images. The client and the point cloud data publisher(by the name *imagepublisher*) can be seen in the ROS node network chart in Appendix B in Figure B.1. The graph visualisation is provided by the *rqt_graph* package.

The software design was modular so that the different parts of the system could be developed independently as modular entities shown in Figure 4.1. The target was to make the software an independent aggregate module that could be used by other researchers in the project later.

In Figure 4.1 the ROS interface can be seen as a dashed line, and the blue

lines show ROS messages that were used for moving data in and out of the ROS network(and between ROS nodes). The orange lines represent UDP packet communication, which was only used in the image data transfer from the cameras to the Windows PC. Different versions of the software can be configured using ROS topics and subscribers: for example, the TLD block can be removed with simple configuration when it is not needed. Dynamic reconfigure server was used to switch quickly at runtime from application to another and enable or disable detection of cylindrical objects.

Besides using standard C and C++ libraries, external libraries were used that specialise in image processing, point cloud processing, file system operations, and communications. *OpenCV* library was used for stereo camera calibration and stereo rectification purposes, and *Point Cloud Library* (PCL) was used for all data intensive point cloud processing. *Boost* library was utilised in all file system operations. All data networking and runtime configurations were implemented using ROS. PCL processing is detailed in Chapter 5.

The version used for all development purposes was *ROS Groovy*. The ROS system core is based on a ROS *parameter server* that keeps track of all topics, nodes, and parameters in the network. The parameter server was implicitly used by all ROS topics, and explicitly used to store machine vision parameters with the help of *dynamic_reconfigure* package. Readily available message types were used for point cloud data networking between selected operating systems.

4.3 Perception Sensor Platform *Himmeli*

Himmeli sensor platform is a modular sensor rig that was engineered in a collaboration of Aalto university, Tampere university of technology(TUT), and VTT in a research project in 2012. The sensor platform was created for use in future automated machinery research and its purpose is to provide flexible sensing of the environment with multiple sensors. *Himmeli* sensor platform was used as the perception platform in this work. It can collect data from the test environments using stereo cameras or a laser scanner, and it can provide orientation change data using an inertial measurement unit(IMU)[74].

The *Himmeli* platform includes three modules(see Figure 4.2): the sensor module, a computation module, and a support module. On the front side of *Himmeli*, the sensor module includes a high resolution stereo camera pair, a single high resolution camera, a thermal imaging camera, an automotive ultrasound RADAR, and an IMU. There is an optional Velodyne HDL-32E LIDAR sensor on top of the sensor module. The Velodyne HDL-32E can be used to scan the environment 3-D geometry with 32 micromirror controlled lasers. The schematic of all the sensors is shown in Figure 4.3.

Table 4.1: Linux PC hardware specifications.

Component	**Details**
OS	Ubuntu Linux 12.10
CPU	Intel i5-3470@3.2GHz
Memory	8Gb DDR3@1600MHz
HDD	500Gb SATA III@7200rpm

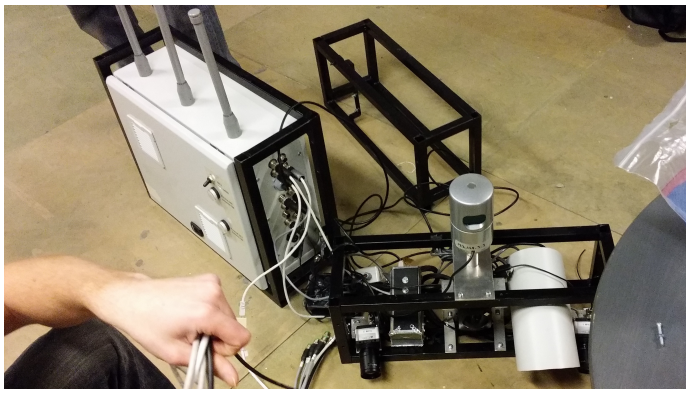

Figure 4.2: *Himmeli* platform includes a sensor module, a computation module, and a support module. On top of the sensor module there is a Velodyne HDL-32E LIDAR sensor.

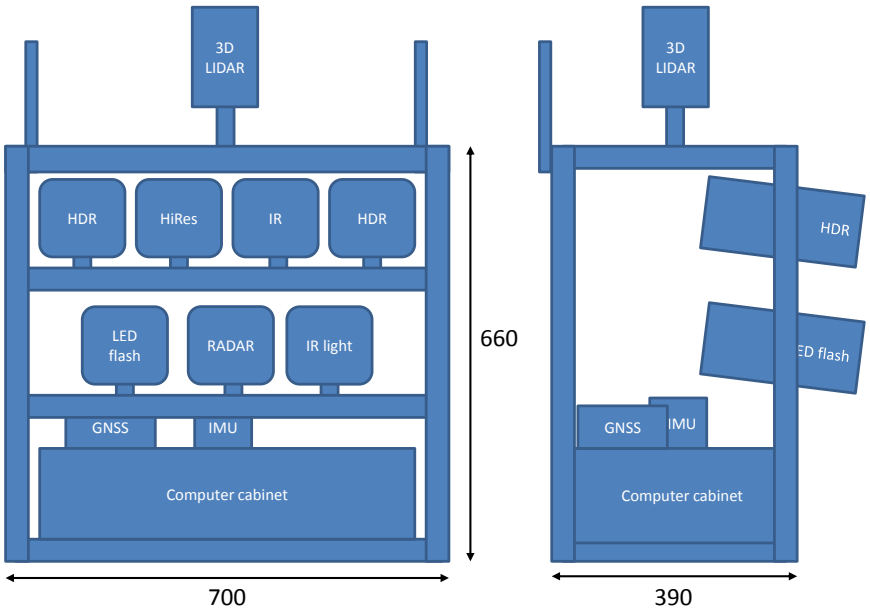

Figure 4.3: *Himmeli* perception platform sensor schematic[51].

The computation unit included a Linux PC for point cloud processing, and a Windows PC for stereo image capture. The images must be transmitted to the Linux PC over a LAN network. The results in Chapter 6 were computed using a desktop PC whose hardware specifications are listed in Table 4.1.

4.3.1 Cameras

The *Himmeli* sensor module includes two high resolution GigE μEye UI-5120 RE HDR cameras with industrial grade casings. The μEye UI-5120 RE HDR camera is a high dynamic range camera with a CMOS(complementary metal oxide semiconductor) sensor array with a resolution of 768 x 576 pixels(PAL). The model in use(UI-5120RE-M-GL Rev.2) has a monochrome sensor that transmits intensity images at a maximum of 50 frames per second(FPS) using Gigabit Ethernet.

If the μEye cameras are installed in a vibrating environment the cameras' image output quality may suffer, because the optics do not include image stabilisers.

The cameras are installed with lens tube shieldings(class IP65/67) that enable operation in outdoor winter conditions. The components of the camera comply to the splashproof and dustproof IP65/65 class, which makes the camera and its casings well-suited for industrial imaging purposes.

The camera CMOS monochromatic sensor has a pixel size of 10 μm and an optical size of 5.75 mm $\times$ 7.68 mm. It has a single channel colour depth of 12 bits, and a logarithmic dynamic range of a 120dB sensitivity[75], which is a larger than standard sensitivity range for a monochromatic camera. Since *OpenCV* processing supports only 8-bit channel depth, the 12-bit channels have been converted into 8-bit channels during the 3-D reconstruction[40], which means that the HDR camera is not used in the most accurate possible way in this work.

4.4 Camera Installation Locations

The *Himmeli* platform was installed in two different test environments in a total of 4 different camera installation locations as shown in Figure 4.4.

Installation location *A* was used in datasets A1-A6. The camera platform was located above the lifted load object at a slight angle of approximately 6°. The perspective seen from the installation location *A* is titled *the top-down view*. In this camera configuration the load object was centered in the camera images at all times.

Installation location *B* was used in datasets B1-B7. The camera platform was installed in the upper side corner of the workspace so that the load object is seen at a 45° angle from the highest possible height in the test space. The perspective seen from the installation location *B* is titled *the bird's eye view (I)*.

Installation location *C* was used in datasets C1-C7. The camera platform was installed at a height of 0.5 meters, and tilted so that the cameras face sideways(the cameras are installed at an angle in the platform). The perspective seen from the installation location *C* is titled *the frog perspective* since the load object is being viewed from below the centre of the load object.

Installation location *D* was used in datasets D1-D5. The camera platform was installed at a height of 3 meters facing the load object at an angle. The perspective

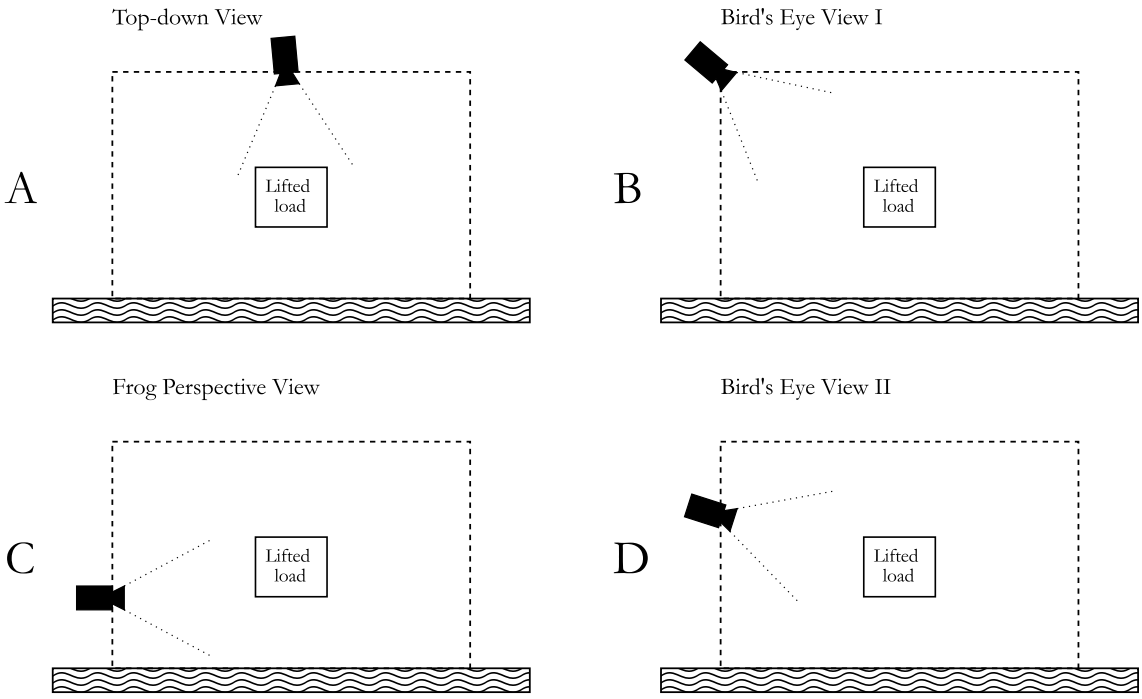

Figure 4.4: Camera installation locations and orientations *A*, *B*, *C*, and *D* used in the test environments. The image shows the lifting equipment workspace with a dashed line. The lifting mechanism of the lifted load object is omitted. The letters A-D correspond with the test dataset names accordingly.

seen from the installation location *D* is titled *the bird's eye view (II)*.

4.5 *OpenCV* Stereo Calibration

The *OpenCV* stereo calibration uses the same techniques in its implementation as the *MATLAB Camera Calibration Toolbox*. The *MATLAB Camera Calibration Toolbox* was first used to find intrinsic camera calibrations for each camera to provide initial guesses for the stereo calibration procedure. The left camera was calibrated using 112 out of 157 calibration images. The right camera was calibrated using 111 of 157 calibration images. The calibration procedure followed a tutorial by Caltech university[76]. For example, the resulting reprojection error in pixels was 0.1797 for x direction and 0.1486 for y direction for the right camera. In general, a result of 0.10 pixel reprojection error is considered a great result, and 0.18 is also a good result. The calibration procedure should produce meaningful results in the 3-D reconstruction phase with reprojection errors less than 0.20 pixels.

The used technique estimated the fundamental and the essential matrices between the two stereo images who contain *calibration chessboard* patterns. The calibration is based on the knowledge of the chessboard pattern crossing locations in

high accuracy. In Figure 4.5, a 9 × 6 crossings calibration chessboard pattern is shown, which was captured in all camera image corners. Additionally, the tilt of the chessboard pattern was increased in all the pattern locations until the *OpenCV* chessboard pattern detector lost the track of the calibration pattern. In Figure 4.5, the calibration pattern is being tilted because orientation data is required for a successful calibration result.

The calibration procedure was highly sensitive to the lighting conditions, and the calibration succeeded better in low lighting than bright lighting conditions in an indoor office environment. The calibration was more difficult in a well-lit environment where specular reflections were seen on the calibration pattern including reflections from the indoor ceiling lighting. The calibration was attempted after installing the perception platform to the installation location A, but it was unsuccessful because the distance to the calibration pattern was more than 4 meters. In testing it was noticed that the stereo cameras must be calibrated before use with a calibration pattern that is nearby the cameras or large enough to cover more image area. The calibration pattern should also be made out of matte finish materials to reduce the amount of specular reflections seen on the calibration pattern surface.

The calibration was successful most often when the calibration chessboard was 0.5-2 meters away from the cameras. The calibration was unsuccessful for all distances higher than 2 meters with the size of the calibration grid that is shown in Figure 4.5. A larger calibration grid must be used if a calibration is attempted from a distance of more than 2 meters with the *Himmeli* camera setup.

Figure 4.5: A 9 × 6 crossings standard chessboard pattern laser printout is being detected simultaneously in both of the camera images. The colour visualisation is provided by the *OpenCV* library function *drawChessboardCorners()*.

Chapter 5

Point Cloud Computation And Algorithms

This chapter introduces computation techniques used to recover the load object measurement from the 3-D data provided by the stereo camera pair. First, preprocessing filters and downsampling of the point cloud are introduced, followed by an object clustering technique. A ground plane removal and a cylinder segmentation are introduced that use parameter estimation techniques. Also, a bounding volume computation is introduced that produces the actual three-dimensional load object measurement. Finally, the algorithms that were written for this work are detailed.

The point cloud data was generated according to the triangulation principle introduced in Chapter 3, and the current chapter presents how the point cloud is processed into a bounding volume measurement with the help of *Point Cloud Library*(PCL). The C++ software source code is not shown, because it is not available for public use. An interested reader can find a large part of the essential source code from the PCL tutorials available at *pointclouds.org* tutorial website[77].

5.1 Point Cloud Operations

Point cloud operations are computed with the *Point Cloud Library*(PCL), which is a fully templated modern C++ library for 3-D point cloud processing. PCL is optimised with *Intel SSE* and *Eigen* library backend, GPU CUDA processing, and parallel programming in order to maximise the computation efficiency on specific PC platforms. PCL installation will require other software libraries, such as *Boost* 1.46, *Eigen* 3.0, *FLANN* 1.7.1, and *VTK* 5.6, which are used as part of the point cloud computation, but not further detailed here.

5.1.1 Filtering And Downsampling

The load object measurement software uses filters to reduce the amount of data in the data pre-processing step. The data reduction is recommended since 3-D data processing requires intensive computing even when optimised. *NaN*- and *Inf*-filters are introduced first.

NaN and *InF* Filtering

The ROS message transportation layer requires an organised point cloud to be used in a ROS message, which is why a *NaN*(not a number) filter and an *InF*(infinity) filter are required to remove invalid values from the arriving organised point cloud. The 768 x 576 grid array of values in the ROS message contains *NaN* values and *InF* values as templates, because the 3-D reconstruction software may not recover a valid depth value for all the pixels. All invalid values are removed using a *NaN* removal filter(*pcl::removeNaNFromPointCloud*) and an *InF* value filter. The *InF* value filter was implemented by iterating through all the data points and removing all points marked as infinity. After using the *NaN* removal filter, the data becomes unorganised, which drops the number of points needed to process from 442 368 points to a lower number of approximately 250 000 valid data points.

Voxel Grid Filtering

A *voxel grid filter* is a geometric low-pass filter that can be used to downsample point clouds that contain a lot of data. For example, the filter can be used to compute lower fidelity representations of the environment(analogous to 2-D Gaussian image pyramids). The amount of downsampling is controlled with a voxel grid leaf size parameter, and a selected type of point reduction method.

The voxel grid leaf size is the parameter that controls the size of the voxel cube length. Currently, the voxels in a filter are strictly cubic, which means that the leaf size parameter controls all three orthogonal dimension cube lengths simultaneously, thus, it controls the number of voxels in a grid and also the output accuracy of the voxel grid.

Currently, the used point reduction method is a center of gravity(centroid, CoG) -based method. In a CoG method, all the points residing inside a single voxel cube are replaced by their computed centroid. The centroid values of all voxels are then output as the result of the voxel grid filter. Using the CoG point reduction method enables a more accurate representation of the underlying 3-D surface than a center of the voxel reduction method[78]. The effect of the voxel grid filter on an original point cloud can be seen in Figure 5.1.

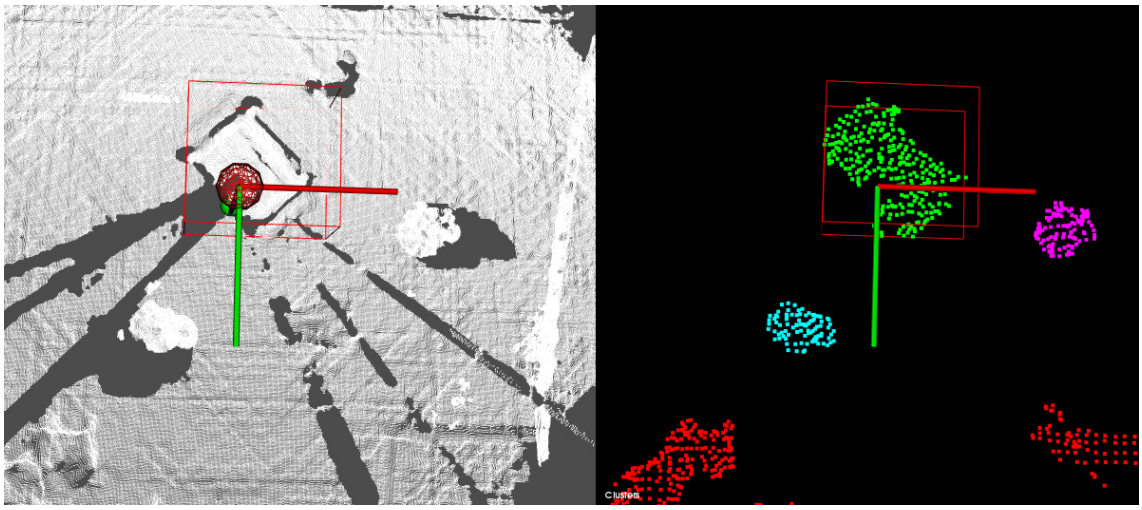

Figure 5.1: A voxel grid filter downsampling of segmented objects. After removing the ground plane, the remaining data points are downsampled to a remaining 2.2% of the original number of data entries. The red box represents an axis-aligned 3-D bounding volume that remains almost identical after data filtering.

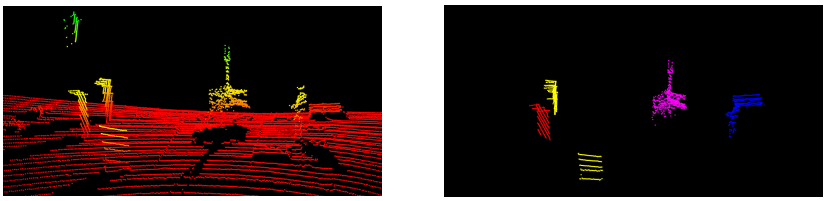

Figure 5.2: a) Velodyne HDL32-E scanner data from the test environment that is used to proof-of-concept the ground plane removal and object clustering technique. b) Segmented object cluster output from a PCL nearest neighbour search.

5.1.2 Object Segmentation

After the preprocessing of the point cloud, the 3-D data is being segmented into separate object clusters using the PCL *Euclidean cluster extractor* as seen in Figure 5.2. The cluster extractor uses k-means clustering technique to cluster n-dimensional data points into k different spatial clusters. For 3-D point cloud data, a Euclidean metric is used in measuring distance between cluster candidates. The extractor is implemented in the *pcl::EuclideanClusterExtraction* class, and the steps of the algorithm are shown in Algorithm 1 that follow the presentation of Rusu's dissertation[79].

Algorithm 1: Nearest neighbour algorithm that utilises a k-D -tree structure.

Data: point cloud P
Result: object clusters C extracted from P

1. create a k-D -tree representation of input point cloud P;
2. set up empty list of clusters C and empty queue Q for points to be checked;
3. **while** *not all points* $p_i \in P$ *have been processed and are part of list of point clusters* C **do**
4. **for** *every point* $p_i \in P$ **do**
5. add p_i to the current queue Q;
6. **for** *every point* $p_i \in Q$ **do**
7. search for the set P_k^i of point neighbours of p_i in a sphere with radius r < d;
8. **for** *every neighbour* $p_k^i \in P_i^k$ **do**
9. **if** *point has been processed* **then**
10. do nothing;
11. **else**
12. Add point p_k^i to Q
13. **end**
14. **end**
15. **end**
16. **end**
17. **end**
18. **return** *object clusters* C

5.1.3 Parameter Estimator For Ground Plane Removal

Before the object segmentation result in Figure 5.1 is possible, the ground plane must be removed from the original data(see ground plane in Figure 5.2a) with a parameter estimator. If the data points belonging to the ground plane are removed, only the load object and other items in the environment will remain in the point cloud data for easy extraction.

Classic parameter estimation techniques(e.g. least squares minimisation) assume that the data used for the parameter estimation does not include outliers. In 1981, Fischler and Bolles presented a new parameterised model estimator titled the *random sample consensus*(RANSAC) that improves classic techniques by detecting which points are inliers in the data[80]. A RANSAC estimator initialises the inlier point search with a small initial dataset and accumulates inliers when consistent data points are found that support a model hypothesis. As the RANSAC iteration progresses, outlier data points are not used in the model fitting, and the solution is found even in the presence of gross errors in the dataset. Because of this, a RANSAC estimator is used to estimate a parameter model for the ground plane from the available test data that contains outliers.

PCL supports RANSAC method with a selection of model types. A model type is required, which in this case is a plane that can be equated with a plane parameter model

$$ax + by + cz + d = 0 \tag{5.1}$$

After randomly picking three non-collinear data points that define a 3-D plane, the model coefficients a, b, c, and d are computed from the randomly chosen candidate plane. Finally, for all data points that fall within a specified error threshold from the candidate plane the distances from the parameter model are computed, and the candidate is scored accordingly[79].

New plane candidate guesses are iterated until a model with small error metric is obtained for the inlier model, or the iteration maximum limit is reached. PCL implementation uses *pcl::ModelCoefficients* class to represent sample consensus model types, such as planar, cylindrical, and spherical implicit parameter models. Moreover, a *pcl::SACSegmentation* class and its methods are used to extract the data points belonging to the found parameter model. A simple tutorial that shows the steps of the plane model segmentation in more detail can be found at *pointclouds.org* website[81].

A typical sample consensus search may run hundreds of iterations, and the number of maximum iterations was limited with a parameter *RANSAC maximum iterations* with a value of 500. A *RANSAC distance threshold* parameter controls the plane error threshold, and the value 140 millimeters was used. Thus, in tests all points that fall within a 28 cm wide threshold area on both sides of the found

plane model are being removed from the point cloud as a result of the ground plane removal process.

If the results from a RANSAC estimator are not satisfying, many variants can be found from the literature[82, 83]. In tests, the computation time of the RANSAC estimator output was on average less than 5 milliseconds for all datasets, and there were less than 50 iterations in the computation, which is why the RANSAC estimator was deemed suitable for processing the datasets. The number of iterations on average per frame was not logged. The computation time of the RANSAC estimator on average was not logged. The computation times depend on the PC used for running the computation, and the number of iterations depends on the 3-D data used.

5.1.4 Cylinder Model Estimation

The cylinder model estimation was used in finding cylindrical load objects from the 3-D point cloud data when applicable. Cylinder model estimation was useful when elongated or cylinder-shaped objects were handled. The cylinder model estimation uses exactly the same technique as was described for the plane model segmentation. The only difference is that the model type is replaced with a cylinder implicit parameter model type. The cylinder fitting technique was used to report the occupied three-dimensional space of elongated objects more efficiently than with the 3-D axis-aligned bounding box bounding volume(that report excessive amounts of empty space).

The cylinder model coefficients in the PCL implementation contain 6 values that represent two 3-D coordinate values, and the radius of the cylinder[84]. The first 3-D coordinate value is a random coordinate found on the cylinder main axis, and the second coordinate is another coordinate found on the cylinder main axis so that the difference of the coordinates report the direction of the cylinder main axis. The biggest drawback of using only 7 values to represent a cylinder is that while the cylinder main axis orientation and translation are recovered, the length of the cylinder is not recovered in this case. Since the coordinates in RANSAC model parameters were never found on the extreme ends of the inlier data, the length of the cylinder must be computed by other means. The solution was to implement a cylinder growing algorithm that projects inlier points on the cylinder main axis and returns both end point coordinates of the oriented cylinder. The algorithm is introduced in Section 5.3.2.

It was not assumed that the load object is found below the end effector tool. On one hand, the assumption could result in a major reduction in the number of data points that must be processed. While this would radically reduce the number of points and lead to faster segmentation of the cylinder model, it can be easily shown that in normal operation the assumption is not valid. The load is often tilted so that it is located above the end effector tool. In order to avoid accidental removing

of the load object feature data, the assumption of a load object located only below the end effector tool cannot be used. Consequently, only object centroid proximity information should be used to decide whether the selected load object is grabbed by the end effector tool or not.

If the simple cylinder segmentation results are not good enough, the segmentation could be improved by implementing a more complex cylinder model that would better approximate the actual shape of the load object(in case a high detail load object model is needed). Such a model that approximates the load object with segments of cylinders is introduced in Forsman's thesis[85]. At this point, a simple cylinder with a linear axis is used, because a more complex object model would significantly slow down the computation and the effect of the accuracy of the model on the bounding volume is minimal, too.

5.2 Bounding Volume Computation

A bounding volume measurement is the target product or information that is produced by the implemented load object measurement software. A load object point cloud is extracted from the 3-D data using the point cloud operations introduced previously. The 3-D bounding volume is computed using either an *axis-aligned bounding box*(AABB) approach, or an *oriented bounding box*(OBB) approach.

AABB is the simplest bounding volume that can be applied in a 3-D Euclidean space. The original definition of AABB is the minimum perimeter along the directions of the axes that span the Euclidean space that fully contain the target object in the 2-D space as seen in Figure 5.3a. It is straightforward to apply AABB in 3-D by finding minima and maxima of the point set along each axis and by spanning a convex polyhedron so that all the minimum values are selected as the other corner that spans the polyhedral graph, and all the maximum values are selected as the opposing corner of the polyhedral graph. This can be done in PCL by using the function *pcl::getMinMax3d()*.

AABB computation is simple, but the biggest drawback of this technique is the large amount of empty space not occupied by the object that the AABB bounding volume contains. In the special case of maximally off-axis aligned cylindrical objects, or planar objects, a 3-D AABB volume representation may report a volume that is mostly occupied by empty space.

An OBB can be used to reduce the empty volume being reported by the AABB. The 3-D OBB rotates its perimeter along the load object main axis as seen in Figure 5.3b. An OBB bounding volume minimises the bounding volume optimally and makes the volume invariant to object rotation in the world frame.

It would be possible to find more complex bounding volume techniques for even more accurate object volume representation, such as the 3-D OBB, bounding volume

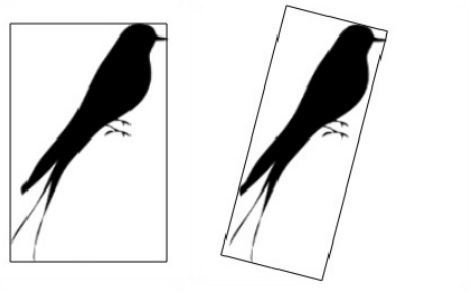

Figure 5.3: a) Axis-aligned bounding box perimeter is aligned along the x- and y-axis. b) Oriented bounding box perimeter is rotated to output a minimum perimeter. The perimeter of OBB is shorter than the perimeter of AABB.

shapes, bounding volume hierarchies(BVH), discrete oriented polytopes(DOPs), k-DOPs and boxtrees[86]. More complex representation of object perimeters can lead to more accurate collision detection techniques. For example, BVH can be used to detect collisions or object interference using raytracing and culling[87].

It is being reported by Ericson that updating of the AABB as the load orientation changes due to rotation is computationally expensive[88], and it is suggested that in order to fight excessive computation, the bounding box should be loosely defined so that with small angle changes there is no need for recomputation.

5.3 Algorithms

5.3.1 Load Object Cluster Selection Algorithm

The load object selection is a difficult problem if the location of the load object is not known. If a location is known, then the problem transforms into selecting an object cluster that is nearest to the load object.

A load selection center is a coordinate used in the k-nearest neighbour search that identifies the load object nearby. To solve the problem of getting the correct cluster *ID* for load object selection, a *load object cluster selection algorithm* was written(see Algorithm 2).

The load object cluster selection algorithm currently finds a nearest point to the selection center for each cluster. It proceeds to select a cluster whose distance is smallest as the load object. Currently, the algorithm could be simplified, but it is left as is to technically support a least squares sum criterion implementation.

Algorithm 2: Cluster selection algorithm that returns the *ID* of the cluster that is nearest to the load selection center according to a simple distance metric.

Data: point cloud clusters C representing segmented 3-D features, load selection center coordinate L
Result: load object cluster ID

initialise vector of float vectors *pointsDistance*;
for *all clusters* $C_i \in C$ **do**
    for *all points* $p \in C_i$ **do**
        initialise a vector of float vectors V_F;
        K = 20;
        compute K nearest neighbours for L from points p and output the result in V_F;
        pointsDistance[i] = V_F;
    end
end
initialise a result vector of pairs of float and integer *results*;
for *all vectors* $v_j \in$ pointsDistance **do**
    sort v_j in ascending numeric order;
    select smallest value $v[1]$ and make a pair $(v[1], j)$, save in *results*;
end
sort *results* according to float values;
load object cluster ID = *resultsDistances*.front().second;
return *load object cluster ID*

5.3.2 Cylinder Growing Algorithm

The PCL cylinder fitting process outputs a parameter set in *pcl::ModelCoefficients* format that contains 7 parameters: 6 of them cover the coordinates of two random points on the main cylinder axis, and the final parameter reports the radius of the cylinder. The starting point and the ending point of the cylinder are not computed at all by the standard RANSAC computation, which is why the *cylinder growing algorithm* was written.

The algorithm projects all segmented cylinder point cloud entries on the main cylinder axis using standard dot product computation, and reports the projected end points of the cylinder. The algorithm runs in approximately 0.1 milliseconds, which is fast for computation of the cylinder length. The algorithm follows standard linear projection mathematics, and the working of the algorithm is presented in detail in Algorithm 3.

5.3.3 Cylinder Segmentation Algorithm

The *cylinder segmentation algorithm* is an algorithm that iterates cylinder model parameters of a cylindrical load object. The algorithm requires information about the load object selection center, which is elaborated next. Possible load object selection centers may be

- the 3-D coordinate point reprojected from a 2-D coordinate acquired by a 2-D image tracking mechanism,
- the centroid of the feature extracted in the previous frame,
- and a default 3-D coordinate value(e.g. origin of a coordinate system, or 3-D coordinate reprojected near the image center).

The centroid is computed here, because statistically the centroid of cylindrical data will be found inside the 3-D surface spanned by data points.

Since the closest object cluster to the selection center is selected as the load object, a selection center inside a cluster will always select the cluster around it as the load object, which is a correct result.

The best selection center is located inside the wanted load object. The next best selection center is located on the surface of the load object. Good selection center is located outside the load object, but it is possible to incorrectly select other objects, too, when a selection center outside the load object data is used.

Selecting the best selection center requires a 2-D image tracker that reports the approximate 3-D location of the end effector in 2 consecutive frames. If the measured 3-D movement of the end effector between frames is smaller than a selected error limit(60 cm) then the centroid value is used. If the error limit is reached, the load

Algorithm 3: Cylinder growing algorithm that returns the end point coordinates of a segmented cylinder for cylinder length computation.

Data: point cloud C representing the load object segmented cylinder
Result: cylinder end point coordinates along main axis

compute a vector A along the main cylinder axis so that all data points in point cloud C project on it;
compute closest data point D to centroid of data;
initialise a start point S at projection of D along the cylinder main axis;
initialise coordinate value H_1 that represents the length of the cylinder height from S to H_1 along the cylinder main axis in direction of A;
initialise coordinate value H_2 at S that represents the length of the cylinder height from S to H_2 along the cylinder main axis in the direction of $-A$;
for *all points* $p_i \in C$ **do**
    compute coordinate p_i projection P_i on cylinder main axis using dot product;
    if *vector* SP_i *is in the direction of* A ***and*** $|SP_i| > |SH_1|$ **then**
        set $H_1 = P_i$
    else
        continue;
    end
    if *vector* SP_i *is in the direction of* $-A$ ***and*** $|SP_i| > |SH_2|$ **then**
        set $H_2 = P_i$
    else
        continue;
    end
end
return H_1 ***and*** H_2

object has moved further away from the previous known centroid, and a new end effector value is used instead.

The selection center is invalidated if

- the end effector 3-D coordinate cannot be extracted from a ROS message,
- or if the end effector coordinates contain *InF*, *NaN*, zero, or default values.

The load selection follows Algorithm 4 for all loads. If the load object is not cylindrical, the cylinder segmentation using RANSAC is skipped, and the dimensions of the load object cluster C_i are reported as is.

Algorithm 4: Cylinder segmentation algorithm that returns model coefficients for correct load object cylinder. The algorithm uses the previously described algorithms, and information about lifted load objects and the fact that load objects are in the direction of gravity from the end effector. The algorithm is simplified, and the while loop terminates in all cases(the number of points in p2r is reduced to 0 or the cylinder is not found by the RANSAC at all).

Data: point cloud P, end effector location E
Result: cylinder model coefficients

initialise cylinder coefficients *coeff*;
initialise cylinder point cloud *cyl*;
initialise candidate cylinder point cloud *p2r*;
initialise cylinder centroid value *cent*;
remove *InF* and *NaN* values from P;
downsample P using voxel grid filter with leaf size 120;
segment point cloud P using k-means cluster extraction, save results in clusters C;
run *load object cluster selection algorithm (Algorithm 2)* on C *and* save result in c;
run cylinder segmentation RANSAC on c *and* save point data in *cyl* and cylinder coefficients in *coeff*;
run *cylinder growing algorithm (Algorithm 3)* on c and save result in H_1 and H_2, save centroid in *cent*;
copy data from *cyl* to *p2r*;
while *!(E projects on the line spanned by* H_1 *and* H_2 *and cent is found toward gravity from E)* and *(number of points in* p2r $>$ *k-D -tree min cluster size)* **do**
    run cylinder segmentation on *p2r*, save cylinder data points in *cyl*, save cylinder coefficients in *coeff*;
    delete data points of *cyl* from *p2r* using data extraction); **if** *number of points in* p2r $>$ *0* **then**
        run *cylinder growing algorithm* for *p2r*, save results in H_1 and H_2, save centroid in *cent*;
    else
        break;
    end
end
return coeff

Chapter 6

Evaluation And Results

6.1 Evaluation Of The Software Design

The design of the load object measurement system evolved a lot during its implementation phase. The original design was a node-based design with atomic operation principle where data would be sent to all the required processing stages using ROS topics and subscribers. Since the amount of data generated in the stereo cameras was quite large, the design had to be changed to a more linear design where the used bandwidth of the ROS network messages per measurement was minimised. The final design included more logical, larger processing components that had many positive effects on the overall performance of the software compared to the first design:

- The overall workflow was more usable and easier to launch (less nodes were launched)
- Less bandwidth was used in the ROS network(less publishers and subscribers needed to be configured)
- Data reduced and information increased per component principle was enforced by the new design

This chapter evaluates the implementation of the software schematic presented in Figure 4.1. The design proved to enable stable heterogeneous PC network communications with the *Himmeli* platform on wired LAN environment in tests. Tests were run in an industrial environment, and also in laboratory conditions. No tests were run in a wireless network environment, but it may be that a wireless network could be a bottlenecking factor due to high data bandwidth requirement of 3-10 Mbit per second(the required bandwidth depends on the number of data points in a cloud).

Many readily available ROS software packages for visualisation, configuration and debugging were used to get the networks running as designed. These utilities proved to be of tremendous help in configuring and troubleshooting the network

settings. The packages used to debug the software during its development were, for example, *rqt_graph*, *rqt_logger*, *rviz*, and *dynamic_reconfigure* packages.

For the selected stereo camera setup and the tested network environments, any amount of data could be relayed from any software component to the next one successfully by using networking with topics and subscribers. No issues were found, although some limitations posed by the architecture were discovered:

- The ROS network introduced varying length delays, as expected
- Custom ROS message content could not be viewed on a subscriber machine that had no message headers built from source code
- Implemented PCL visualisation introduced random periodic delays of more than 1000 milliseconds

All the ROS installations used in the PC network were running on ROS Groovy, and no compatibility issues were found in the bounding volume measurement system network even when different build environments were used(*catkin* and *rosmake*). Different versions of the ROS distribution are not cross-compatible, thus, ROS Groovy was selected as the version to be installed on all systems.

The biggest risk in the beginning was missing ROS knowledge in the project team, and it was known that the ROS programming learning curve is steep for beginners. The missing knowledge risk realised as long development time of the system, but as such it was an acceptable risk to take. Since ROS is a community-based effort to provide software nodes for robotic programming, a well-maintained index of all the available content did not exist, and it was often difficult to find the right tools and learn how to use them. Tutorials and examples were used to implement all ROS node functionality, and some features, for example the dynamic reconfigure server(*rqt_reconfigure* package), suffered from unstable functionality at the time.

Then again, the PCL library documentation proved to be excellent, easy to navigate, and well-documented. All the point cloud processing features were tested using PCL tutorials, which helped a lot in understanding point cloud computation. The code was further advanced with the help of the PCL API documentation. All point cloud processing functions are optimised for fast processing of data, and interactive visualisations were easy to implement using the PCL visualisations library.

The only problem in the PCL design was its visualisation functionality introducing fairly periodic but still random delays of 1000-4000 milliseconds. This may be due to large amounts of memory reserved and freed periodically, but a further analysis of the source of the delay could not be done in the time available for software development.

A suggestion for improvement regarding the software design would be to run memory management and process thread bug testing for the software to check for

errors arising in continuous operation. Another suggestion would be to branch out a version of the software where the visualisation may be disabled after the machine vision system is properly configured. This would effectively make the software faster, because it takes most time to render point clouds for the visualisation window.

The evaluation of the load object measurement system was done using offline datasets that were recorded from actual lift equipment environments. The results from different experiments are presented in Section 6.4.

6.2 Evaluation Of 3-D Data Quality

The point cloud data quality was affected by the overall machine vision process including stereo calibration, disparity tuning, and the hardware setup. The hardware setup and the stereo calibration were used as monolithic initial calibration items. The calibration was done automatically for the hardware setup using *OpenCV* as described in Section 4.5, and after a good basic point cloud output generation was achieved, no further engineering was done for the hardware or the stereo calibration. After an initial point cloud output was generated, the problem of understanding the data quality could be presented: how accurately does the point cloud data resemble the real-world features that it is describing?

The generated point cloud resembles the real scene, but some noisy parts or parts that are difficult to model do not look like their real-life counterparts. Accuracy of the point cloud representation can be quantified using point cloud data *quality indicators*. The most important indicators in the point cloud data quality assessment are

- location accuracy of the surface,
- surface noisiness,
- number of speckle regions, misinterpreted z-depth regions, and artefacts of any kind,
- and the number of finite data point entries.

Point cloud data quality indicators are used in the modeling of urban constructed environments and in airborne LIDAR mapping. In these research fields, the quality evaluation is done using two different goodness analysis mechanisms: *spatial structural analysis* and *positioning analysis*[89]. The need for different quality indicators depends on the application. Since the test environments used in the 3-D modeling are constructed industrial environments, it is a good idea to try to apply spatial structure analysis in this work. Positioning analysis is not needed in this work, because multiple-view registration is not used.

Additionally, reverse engineering of shape primitives from point cloud responses[90] have been used for point cloud data quality verification in industrial settings[91].

Often many box-shaped or cylinder-shaped objects can be found in industrial environments. If their size is known, they can be used for 3-D geometry verification purposes. For any primitive geometric shapes it is fairly simple to quantify a goodness of fit value using geometric fitting e.g. least squares error minimisation.

In spatial structure analysis, some real-world feature qualities are measured from the environment, and the same qualities are then computed from the point cloud response. Such qualities are for example: a location, an orientation, a volume, smoothness of surface, or some other attribute of the feature that can be recovered from the point cloud data. The actual and computed measures are compared and it is possible to determine a goodness of fit value that can be further processed into a goodness of data quality information using a selected criterion. Formulation of a meaningful goodness indicator may be difficult in the presence of missing surfaces, measurement noise, and increasing depth estimation error with distance among other problems with a point cloud representation.

A single effective spatial structure analysis that was possible to do with the captured offline data sets was to check how perpendicular was the angle between standing cabinets and the floor in the datasets. For example, a cabinet is known to form a 90° angle with a floor, and if the scene generates a point cloud response where the angle is only 85°, some understanding of the geometry error tolerance can be formed. An example can be seen in Figure 6.1 where linear features are verified against line segments. The result in Figure 6.1 is from dataset A1, which seems to maintain the geometry well. This will give indication of goodness of structural integrity only for the local neighbourhood of the feature in the point cloud. In this case, the good structural integrity in the near-field feature does not guarantee similar accuracy for far-field features.

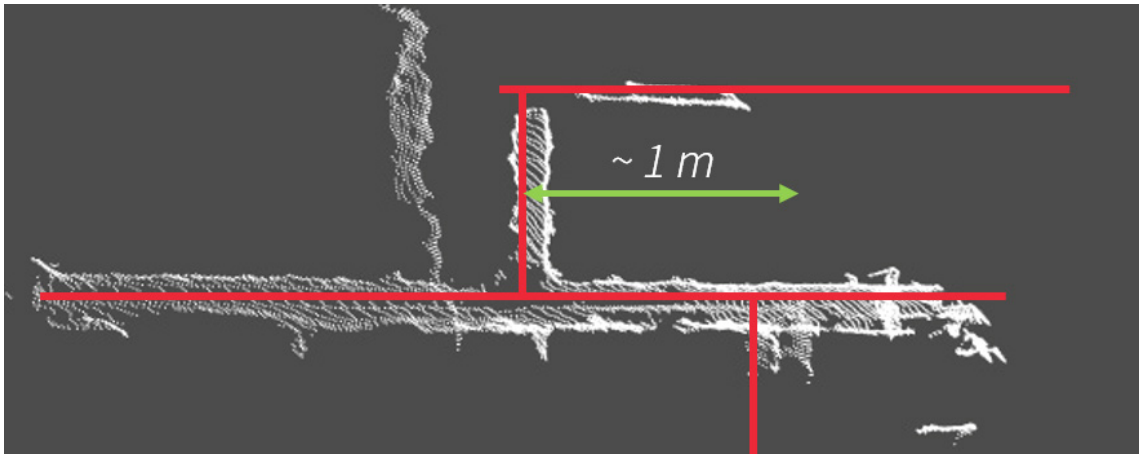

Figure 6.1: Spatial analysis of dataset A1 shows well-maintained geometry in the near-field of the camera. The far-field features introduce more noise in the data and do not present such good results in a spatial analysis.

The effects on the 3-D data quality of a stereo camera setup that has converging optical axes were not researched. Still, a converging camera installation was used in this work, and hardware does affect the 3-D point cloud quality. It is known that converging cameras introduce some vertical disparity, but the effect it had on the point cloud quality was minor.

One thing that was noticed after a recalibration of the stereo system was that the stereo camera pair angle of convergence affects the number of points in the generated point cloud output. With increasing toe-in angle, the size of the region of interest(ROI) in the rectified image decreases. The lower resolution could be seen from the calibration files that reported the ROI of the rectified image in pixel values. *OpenCV* did take the changing toe-in angle into account as the system was recalibrated after changing a camera angle, but as the rectified image became more and more distorted with the increased toe-in angle, the grid array available for the 3-D point reprojection became smaller. Accordingly, the number of points in the 3-D point cloud decreased.

6.3 Block Matching Tuning Evaluation

The block matching algorithms in *OpenCV* need tuning of 10 parameters that are listed in Table 6.1. The best parameters depend on the application and the conditions of the environment, thus, they should be easily changeable during runtime. The parameter values can be chosen arbitrarily within the ranges stated in Table 6.1.

A custom software tool was programmed that displays a set of sliders, which control each parameter separately. The tuning tool paused the stereo camera frame so that it was possible to detect the effect of a change of a single parameter in the same 3-D frame. As the user changes a value of a single parameter using a slider, the 3-D point cloud visualisation is being updated instantly.

The load object measurement software currently supports two block matching algorithms, the *block matching* algorithm(BM) and the *semi-global block matching* algorithm(SGBM) that are supported by the *OpenCV* library. The SGBM algorithm includes additional smoothness constraint parameters p_1 and p_2 whose effect on the point cloud output can be reviewed in Figure 6.2. On one hand, a large value for p_1 and p_2 will result in a large number of points even in low-texture image regions, but a smoothing effect of the data was noticed. On the other hand, a small value of p_1 and p_2 resulted in a smaller number of points, but there was no undesirable smoothing effect.

The tuner was the most helpful aid in creating the load object measurement software since it helped study effects of a single parameter for a scene. For example, the effects of the *fullDP* parameter was easy to study: the *fullDP* parameter runs a

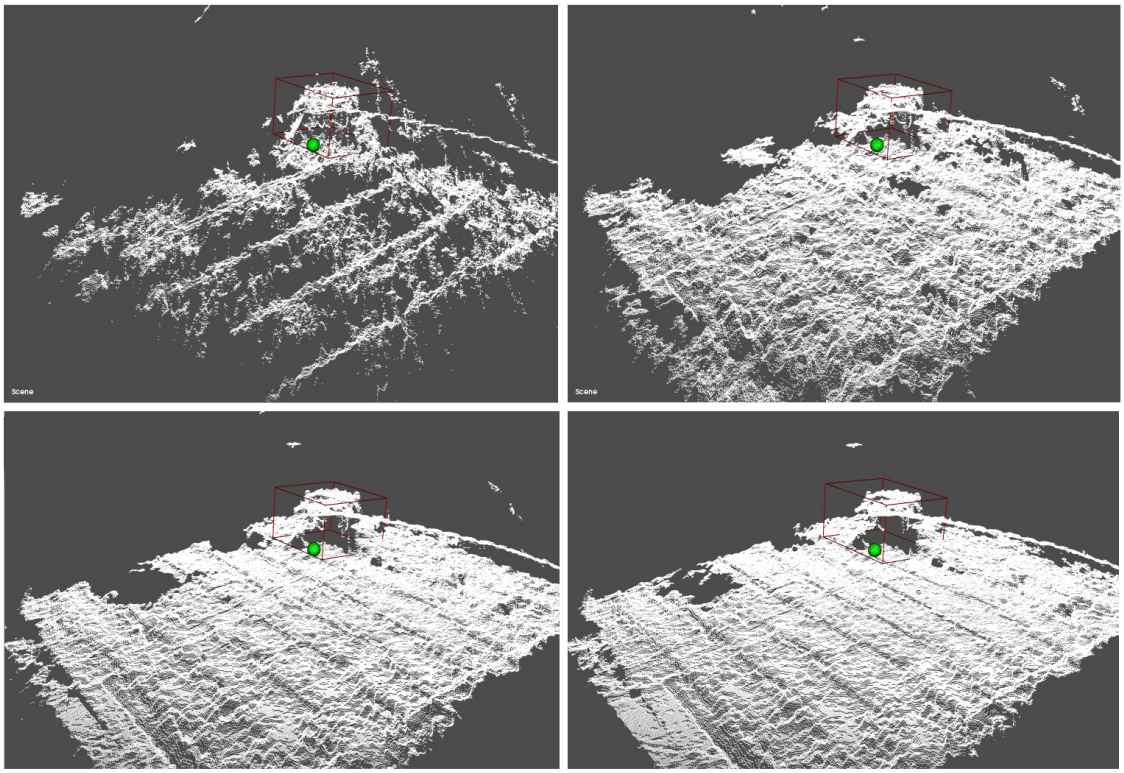

Figure 6.2: The effect of parameters p_1 and p_2 in the SGBM algorithm tuning. Values of p_1 and p_2 increase from left to right, and from top to bottom.

full-scale two-pass dynamic programming algorithm, which consumes large amounts of memory. The slider was used to switch the dynamic programming on and off, and the difference in computing speeds and point cloud quality could be seen instantly in the point cloud output.

The tuning of the BM parameters was done using datasets from all camera configurations A, B, C, and D. High saliency(high frequency intensity change) image regions were used for visual inspection of the tuning, because such regions produced good point responses. For example, the load object inside the red box in Figure 6.2 was generated from a high salient image region with vividly changing textures.

6.3.1 Block Matching Parameter Effects In Detail

First, the tuning was started with a coarse selection of parameters to find disparity that approximately resembles the correct ground truth disparity. Especially, the sum of absolute differences(SAD) window size was kept higher than normal to produce a lot of visible matched points at first. The most important parameters in the final BM state tuning set are SAD window size, number of disparities, and speckle range parameters. These parameters should be tuned first, and other parameters can be

used for fine tuning of the disparity map afterwards.

The **SAD window size** affects the size of the sliding kernel window used for the correspondence search in the 1-D epipolar search space of the rectified images. The smaller the value, the smaller the sliding smoothing kernel used is, and with a minimum value the search is called *pixel-to-pixel correspondence search*. For metric measurement purposes the disparity map should be as accurate as possible, thus, the minimum value of the SAD window size will be used. If the rectified images are texture rich, then a match can be found for almost all pixels on an epipolar line. Accordingly, if the image pair is less rich in texture(e.g. showing a matt painted floor) the process will find less matches. A low-texture environment will produce little found matches, because the local neighbourhoods of the sliding window are too similar in order to confirm a match. If the point cloud is used for 3-D visualisation purposes, a higher SAD window size value selection should produce more points.

The **number of disparities** parameter sets the maximum disparity difference in pixels for a search space. The higher the value, the higher a pixel difference value is allowed in the disparity map. For example, if a near-field object has a disparity difference of n pixels, and number of disparities property is set smaller than n, then the near-field object is not detected at all in the disparity map leaving default values in it. This property can be used to remove near-field occluded objects that cannot be triangulated successfully. With number of disparities parameter it is possible to remove these objects from the disparity search space, and altogether from the final point cloud.

The **speckle range** parameter controls speckle, which are disparity patches generated by the block matching algorithm near object boundaries. The block matching sliding window will catch object foreground and background in the image pair, which generates speckle to the disparity map. The speckle range controls the threshold for letting the local patches of speckle be matched to the resulting disparity map. If the threshold is met, then the local region is eliminated and no disparity matches will be available for that region.

The rest of the parameters, namely **pre-filter size**, **pre-filter cap**, **texture threshold**, **uniqueness ratio**, and **disp12MaxDiff** should be tuned after the previously discussed parameters. The tuning effort resulted in the best possible BM and SGBM parameters for each dataset. BM parameters that were used in the datasets A1-A6 and B1-B7 are listed in Table 6.2. The BM parameters that were used in the datasets C1-C7 and D1-D5 are listed in Table 6.3.

6.4 Results: Findings From Datasets

Up to 25 different datasets were available for analysis, and the findings from all datasets are presented first. The datasets are separated into indoor datasets with

Table 6.1: List of parameters in a block matching state object and their value ranges and requirements in *OpenCV* C++ image processing library.

Parameter name	**Minimum**	**Maximum**	**Required**
Pre-filter size	5	21	Odd number
Pre-filter cap	1	63	
SAD window size	5	255	Odd number
Minimum disparity	-100	100	
Number of disparities	16	256	Divisable by 16
Texture threshold	0	no upper limit	
Uniqueness ratio	0	255	
Speckle window size	0	100	
Speckle Range	0	100	
Disp12MaxDiff	0	no upper limit	

Table 6.2: Block matching parameters used in the datasets A1-A6 and B1-B7.

Block matching parameter	**Value**
Pre-filter size	9
Pre-filter cap	63
SAD window size	5
Minimum disparity	0
Number of disparities	96
Texture threshold	270
Uniqueness ratio	20
Speckle window size	42
Speckle range	10
Disp12MaxDiff	10

Table 6.3: Block matching parameters used in datasets C1-C7 and D1-D5.

Block matching parameter	**Value**
Pre-filter size	9
Pre-filter cap	63
SAD window size	5
Minimum disparity	0
Number of disparities	144
Texture threshold	30
Uniqueness ratio	1
Speckle window size	42
Speckle range	14
Disp12MaxDiff	1

camera viewpoints A and B, and into outdoor datasets with camera viewpoints C and D. Most datasets proved useful in realising what the difficulties of implementing a machine vision measurement system that measures a three-dimensional volume are. The indoor test dataset findings are discussed first, and the outdoor test dataset findings follow. Finally, detailed analysis for datasets A1 and A6 are presented.

6.4.1 Indoor Measurements

All the analysed indoor image datasets were captured in an industrial test environment in August 2013. A total of 13 indoor datasets(A1-A6 and B1-B7) were captured that contain different tests revealing how the dimension measurement of the load object can function in different scenarios. Environmental conditions did not change during the tests, which is why all the test results are comparable in datasets A and B.

A manual load selection approach was used since no end effector location service was available at the time of dataset recording. Instead of using image tracking, a pre-defined coordinate was selected that chooses the nearest object as the load object. The load object selection center was set in the center of the captured image where a load object usually appears in e.g. the top-down camera A. This kind of proximity-based load object selection introduced a lot of erroneous load object selections, such as people walking nearby the center of the image, or new objects selected as they came in the view. The decision to use a proximity-based load object selection instead of a 2-D image tracker was far from perfect, but in the end no information about the end effector location was required to select and measure the load object, which was good. Wrong selections cause visual clutter in the dataset visualisations in Appendix A, which is caused by sequentially changing objects that report differing measurement values. The proximity-based selection was detailed in Algorithm 2 in Section 5.3.1.

The coordinate system was not transformed to the world coordinate frame in any of the datasets, but kept in the sensor frame during the data processing. The sensor frame was used for processing because the sensor frame implicitly encodes a moving platform signal as part of the data. The effect of the embedded platform movement signal can be seen in Figure A.2 in Appendix A where in the upper right corner(object point cloud maximum value) e.g. x-axis signal generates a slope in the time series. The slope spikes up to a new value in a saw waveform like signal every time the load object being tracked moves too far away from the load object selection center and a new object is being selected. It can be seen from the upper left corner of Figure A.2 that during a slope the AABB bounding volume stays practically constant, and as a spike in the maximum coordinate value is detected the AABB bounding volume switches to another object and new constants are being introduced.

The effect of the fluorescent lighting in the test environment was not analysed, and the lighting conditions could not be controlled. The tests were done in 50 Hz fluorescent lighting conditions. The fluorescent lights have some effect on the image quality, but it is unknown how the HDR camera parameters change in different lighting conditions. The effects of lighting in the image quality cannot be assessed without better knowledge of the HDR camera image formation mechanism. The effects of lighting conditions on the image quality in a machine vision solution are a vast research topic on their own and not elaborated in this work.

Findings From The Top-down Viewpoint In Datasets (A1-A6)

From **dataset A1** it was found out that the items residing in the test area are segmented in good detail as separate object entities. The geometries of items are well-preserved when the top-down camera configuration A is used. Accordingly, the objects in the near-field of the camera are surface reconstructed in fine detail by the stereo vision system. Finally, the disparity map includes some artefacts that generate erroneous 3-D constructs in the point cloud data. The best example of an artefact is the tall curved bar that can be seen to the right from the center of the image in Figure 7.2. The bar can also be seen in the disparity map, which is why it is being reprojected as a curved tall beam-like feature in 3-D data point cloud output. This artefact could be effectively removed by using a smaller ROI before reprojecting the data into 3-D.

Another issue found in the geometry reconstruction was caused by a horizontally aligned striped black and yellow tape, which resulted in a depth measurement error. Although the tape was part of the floor and should have been removed from the 3-D data in the process, it remained in the data as a separate object seen below the floor level because of unsuccessful block matching.

From **dataset A2** it was found out that objects suffer from enlarging effect due to disparity map processing. The enlarging effect will affect groups of objects placed near each other the most, causing fusing of objects into single object entities as seen in Figure 6.3. The enlarging effect is typical in a stereo triangulation solution that uses disparity mapping, and it is caused by the image kerneling used in the block matching(e.g. a 3×3 kernel introduces some fattening effect) and mathematical problems in the disparity map estimation.

From **dataset A3** it was found out that a human and the load object will merge as a single point cloud entity when the human stands near the load object or touches it. In the event of a person and the load object fusing, the bounding volume will become too large accordingly. In general, any objects will merge as one when the Euclidean distance between the two objects becomes smaller than the threshold value used in the object cluster segmentation. This suggests that a collision can be detected by detecting large changes in the bounding volume measurements. For

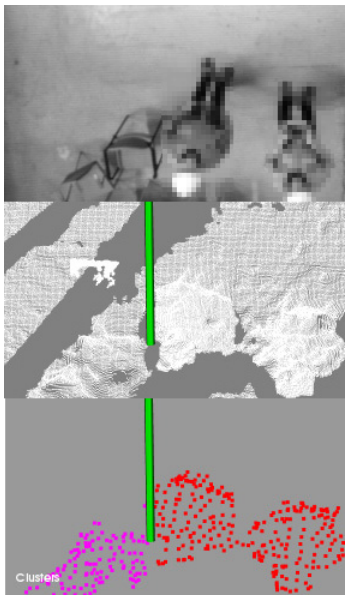

Figure 6.3: Enlarging effect seen in dataset A2. People are actually separated with a gap between each other, but the segmented object view shows the two humans fused as one red point cloud object due to enlarging effect in the disparity mapping processing.

example, in dataset A3 the lifted load object collides with a standing cabinet. It can be seen that a spike in the AABB bounding volume is caused when a collision happens. In Figure A.3 in Appendix A the y-axis large values in the frame range [400 500] are caused by a collision of the load into standing cabinets.

Finally, when the load object was lifted high up near the stereo camera platform, it occluded most of the floor visible in the image. If the floor is not visible in the left camera image, the ground plane segmentation step will fail. The event of ground plane removal failure was not encountered in any of the datasets although there is a possibility that the ground plane will be fully occluded. To sum up, the camera platform installation location A above the load seems to work well in spite of occlusion of floor when lifting the load object.

In **dataset A4** the cameras were moving while the stereo images were captured, which introduced problems in the estimation of the ground plane depth. It was noticed that the ground plane geometry showed large depth projection errors in frames where the cameras were moving. It was concluded that since the stereo camera aperture shutters are strictly not enforced to expose the scene at the same time, the platform has time to move before the second camera shutter is being activated. This results in an increase or decrease in the stereo baseline width b randomly between the frames(depending on which stereo head exposes the image first). The changing baseline will result in erroneous depth computation for all

motion field vectors(moving parts of the image). This also implies that only the moving parts of the image are affected, for example, the ground plane depth values are highly affected. The load object depth measurement is not affected since the image of the load object is not moving relative to the moving cameras. To counteract inaccuracy caused by a moving platform, the cameras should not move while the environment is being modeled, or the shutter behaviour should be synchronised with a clock generator signal.

In **dataset A5** it was noticed that when people walk exactly under the camera view in a top-down camera setup, the point response for a person will become so small that it is discarded as noise in the point cloud data. On one hand, this result suggests that the limit of the point cloud object size should be made smaller than the currently used value(80 data points) to be able to detect small targets. On the other hand, making the limit smaller will allow small disparity speckles clutter the 3-D data as objects. Since the original point cloud is downsampled using a voxel grid filter, such a scenario may not be problematic at all, because noise speckles are filtered away from the downsampled point cloud. Similarly, in dataset A3 the small end effector used gives a small point response of approximately 60-80 data points when the end effector is lifted 2 meters above the floor. The end effector response is so small that it is discarded from the segmented data as noise with the current segmentation tuning. The end effector response could be tracked with an *extended Kalman filter*, because the small response is consistent throughout the dataset.

Findings From The Bird's Eye Viewpoint In Datasets (B1-B7)

A bird's eye viewpoint is introduced in the **dataset B2**. It was found out that the load object being lifted near a wall results in a smaller point response than the load object in the same spot using the top-down image. It is also likely that a load object situated near the wall will fuse with the wall point response. The load object could not be measured if it merged with the wall. The more distance to the cameras, the more easily merging of objects happens since the measurement error increases, and points will easily deviate from their actual values and cause merging. The distance to the wall was quite large, even 7 meters, which caused less accurate depth estimation and object merging.

Another problem with the bird's eye viewpoint was that the backside of the load object was not visible at all, thus, the backside of the load object was not inside the computed bounding volume. This is especially problematic when using camera installation location B and D.

It was found out from **dataset B3** that the 3-D model of the scene includes many more data points when the cameras are in a bird's eye configuration. This is because the large wall is visible in the stereo image pair, and it generates a lot of points. It was also noticed that the computation time increased considerably

because of the increased amount of data remaining in the point cloud.

In **dataset B4** the ground plane was not fully removed by the RANSAC estimation procedure at the furthest measured location. This is mainly induced by the disparity mapping processing of the interface of the floor and the wall where a sharp 90° turn is not being well-modeled. In reality, the disparity map can only approximate sharp turns with smooth trajectories, thus the floor seems to curve up and continue as a wall section. The modeling gets less accurate with increasing distance to the stereo camera platform, and at 7 meters distance the smooth curving effect is so large that the ground plane is not fully removed because of the curving interface near the wall section. Still, the ground plane estimation works flawlessly, and the curving effect does not cripple the RANSAC estimator.

Figure 6.4 shows the problematic curving of the interface of the wall and the floor, which is generated by image regions below the center of the image(the cameras are installed upside down in camera installation location B). According to Figure 3.3 the lens distortion is not large in the lower part of the image, which means that the effect is mostly caused by the disparity map smoothness constraint. Section 3.10 detailed the disparity mapping and its trade-offs in more depth.

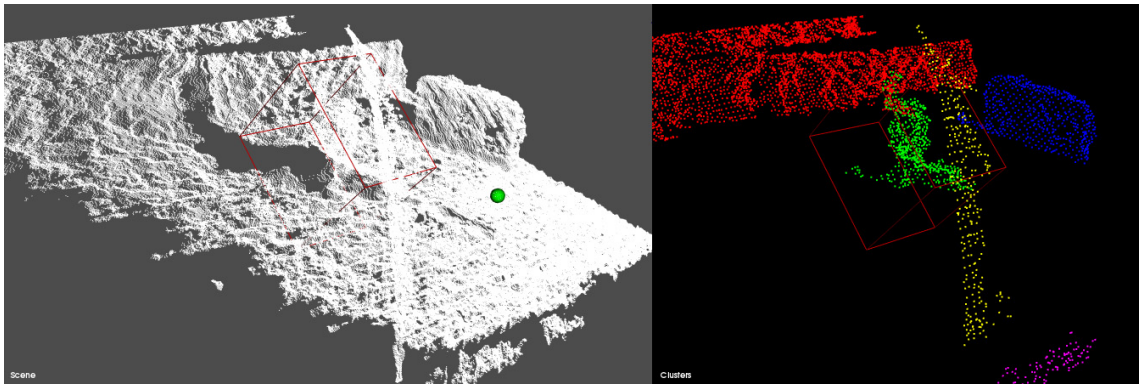

Figure 6.4: A point cloud generated from the indoor dataset B4 shows the curving of the interface of the floor and the wall. The wall section can be seen as the segmented object coloured in red. The artefact seen in most 3-D point cloud representations in this work can be seen as the segmented object coloured here in yellow.

Dataset B5 proves that it is possible to lift the load object fully out of the stereo camera image if the load object is located far away in the bird's eye view. If the load object is located near the wall, the load object will easily merge with the curving floor section even though the load object is already lifted high in the air. Finally, when the load object was brought near the cameras, the visible surface model of the load object was recovered accurately with high resolution. The accuracy of the stereo reconstruction depends on the distance to the load object and the effective image size.

In **dataset B6** some partially see-through objects were measured as a test. It was found out that partially see-through items cannot be stereo matched, thus, they are invisible to the machine vision system implemented. Additionally, shiny metallic materials may have an effect on the stereo matching processing since the surface violates the assumption of the Lambertian reflectance model.

Another remark from this dataset was that upright cylindrical objects can be easily modeled from the bird's eye view used. Top-down cameras have difficulty on modeling other than the top part of an upright cylinder. Such objects include road construction cones etc. It is easy to generalise that objects whose main axis is perpendicular to the camera optical axis are modeled in better detail than objects whose main axis is in parallel with the optical axis of the stereo vision system.

In **dataset B7** two non-reflective large markers were added on the see-through load object. The load object then generated two visible point response speckles as the output. Still, measuring a geometry of an object that is spanned between two marker point responses is currently not supported.

6.4.2 Outdoor Measurements

A total of 12 outdoor test image datasets were captured from an outdoor test environment in August 2013. An automatic tracking-learning-detection(TLD) tracker was used in combination with a cylinder fitting scheme to provide more meaningful results for measurements. The TLD tracker provided the location of the end effector tool with good accuracy using 2-D image feature tracking[92]. The C++ implementation of the TLD tracker is described in Nebehay's thesis[93]. A simple cylinder fitting scheme was used according to Section 5.1.4. It is good to understand that some of the findings in this section are exclusively based on the cylinder fitting results and the implemented load selection algorithm that was further detailed in section 5.3.1.

Findings From The Frog Perspective Viewpoint In Datasets (C1-C7)

It was found out from **dataset C1** that objects too close to each other will merge into single objects similarly as in the indoor case. The merging of objects can be controlled using the *k-D -tree search cluster tolerance* parameter, which affects the segmentation threshold distance. Occlusion made the problem of object segmentation more difficult to solve since usually the occluding object will fuse with the occluded object given that the objects are nearby each other.

It was noticed in **dataset C2** that if the end effector tool is empty, the cylinder fitting procedure does not produce meaningful results as can be seen in Figure 6.5. Currently, the cylinder fitting will result in the volume of the end effector tool or parts of the visible boom depending on where the RANSAC search finds a cylinder fit

in the data. This behaviour adds noise and erroneous measurements in the bounding volume measurement visualisations of outdoor test results.

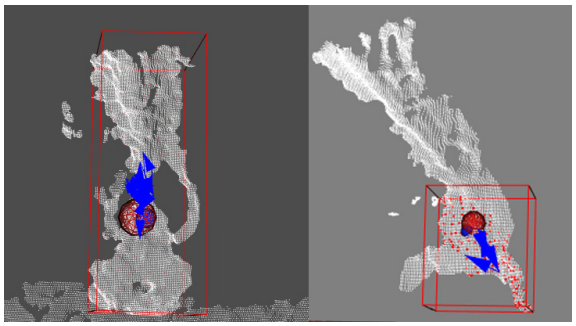

Figure 6.5: An empty end effector tool does not produce meaningful results when it is used with the cylinder fitting computation. The resulting output may report the dimensions of a) visible part of the lifting mechanism b) parts of the end effector that fit a RANSAC model well enough.

A rotation of the object perpendicularly to the optical axis will report different bounding volumes for different angles of rotation in **dataset C3**. The measurement was most accurate when the maximal length of the object was shown in the image. Let this angle be the 0° angle. A rotation of 0-65° will give similar results, but a rotation of more than 65° will cause decreasing image area of the object and generate a poor object surface model. In the case of 90° rotation(self-occlusion) only the other end of the object is visible, and in this situation it was not possible to measure the actual dimensions of the object at all. The measurements produced by a rotating cylinder can be seen in Figure A.7 in Appendix A. It can be seen from the bounding volume chart that as the object is being rotated, the three-dimensional measurement is a function of the orientation of the load object. The missing measurements were caused by the TLD tracker that loses the load object detection when the load object is being constantly rotated.

It was noticed that high smoothness constraint of the disparity map in the SGBM algorithm fills holes that are found in the environment. For example, the loops formed by the wiring seen above the end effector tool in Figure 6.5a will be filled when the SGBM algorithm is used(see Figure 6.5b). The hole-filling phenomenon is present in the SGBM algorithm implementation, but not in the BM algorithm implementation, because there is less fattening effect by the disparity map. It was concluded that while the SGBM algorithm produces better 3-D models of the environment with more data points in the output, the BM algorithm actually can have finer detail in some features with faster computation speeds. In the end, the SGBM algorithm was more suitable for the development of the prototype software. The algorithms can be used interchangeably depending on the dataset and suitability of each algorithm to each situation.

The cylinder fitting can handle picking of multiple objects at once if a single cylinder approximation is used. A single cylinder approximation means that multiple cylindrical objects are modeled as a single large cylinder. Multiple objects tend to be in an hour-glass shape more than in a cylinder shape. Because of this, the cylinder fitting can fit most volume of the objects inside a single cylinder. The furthest ends of multiple objects may stay outside the bounding volume as can be seen in Figure 6.6. It is possible that the enlarging effect of the disparity map generation increases the bounding volume so that the objects actually stay inside the reported bounding volume. It remains to be verified whether the actual objects would fit inside the computed bounding volume or not when using a single cylinder approximation. In dataset C7 three cylindrical loads were lifted simultaneously, and the data resemble single cylinder measurement results(see Figure A.9 in Appendix A).

From **dataset C4** it was discovered that when the crane end effector is brought in the proximity of objects on the ground it is possible to measure a single object dimension before the object is grasped by the end effector tool. Measuring loads from the ground is only possible for objects in the close proximity of the end effector tool. This phenomenon is useful in speed-up of the load object measurement.

In **dataset C5** an object was picked up from a pile of objects. All the objects in the pile form a single point cloud from which the detection of the wanted load object must be done. Figure A.8 in Appendix A shows a 3-D AABB response of a pile of objects. It is similar to a single cylindrical load object, but it is slightly enlarged by the surrounding pile of objects. The cylindrical load object is found purely by utilising RANSAC estimation. In general, if the measured object is occluded by other nearby objects, the cylinder fitting will easily return wrong matches without heuristics being used to guide the correct selection. It was noticed that it is especially easy to fit a cylinder on a person walking nearby. This could be solved with heuristics, for example, a good heuristic rule to use would be *cylinders that are in 90° upright position in the world frame are not load objects*. Humans walking in the area are not easy to distinguish as separate objects if they are nearby load objects. For example, a person walking over a pile of objects will be grouped as part of the entity that represents the pile of objects.

It was also noticed that the load object cylinder fitting introduces errors because the backside of the cylinder is missing from the 3-D point cloud data. A 3-D reconstruction only generates a visible surface model, which means that the cylinder RANSAC computation estimates the cylinder model parameters using partial data. The outcome may or may not describe the actual cylinder, but the cylinder translation is incorrectly estimated as too near the cameras. Missing backside of the object does not affect the orientation estimation. The estimated position, orientation and volume of the object are useful, but the estimates could be further improved if another camera viewpoint was registered into the original point cloud data.

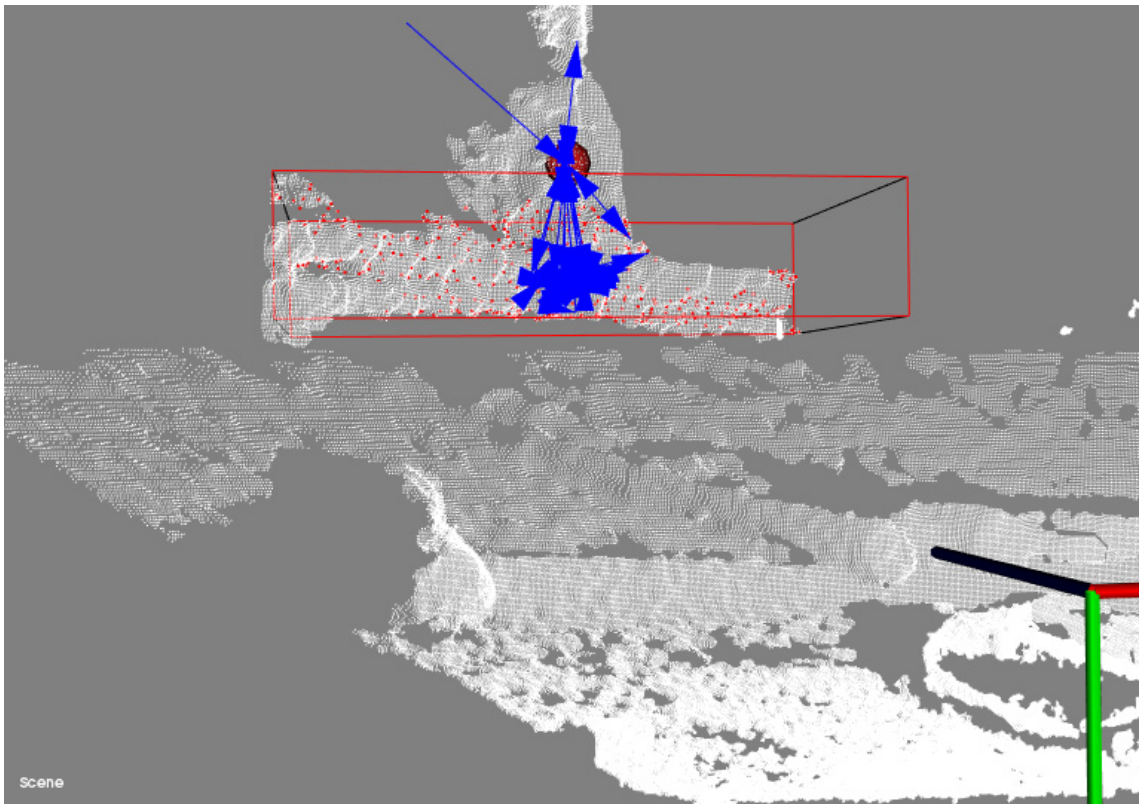

Figure 6.6: Multiple cylindrical loads are modeled using a single cylinder fitting. A single cylinder approximation results some parts of the multiple cylinders being outside the AABB bounding volume. Multiple cylindrical objects tend to form an hour-glass shape formation. The blue arrows depict a history path of the location of the load selection center.

Findings From The Bird's Eye Viewpoint Datasets (D1-D5)

It was noticed from **dataset D2** that it is possible to measure the current state of the end effector tool from the 3-D geometry data. It is possible to report whether the end effector is closed or open, and whether it carries a load or not. In some tool orientations the end effector state estimation is an easy problem, but for some orientations the problem can become difficult. For example, in Figure 6.5 it can be clearly seen whether the end effector tool is in a closed or open state.

It was discovered in **dataset D3** that a partially occluded object only reports the geometry measurement for the visible part of the object. For example, if 50% of the object is visible in the image, then approximately 50% of the actual length of the object is reported by the measurement system. Occlusion is typical in situations where the object has moved behind a lifting mechanism, and the occluding object is the lifting mechanism itself.

Finally, the **dataset D4** simulates normal workflow using the lifting equipment.

It can be seen from Figure A.10 in Appendix A that the 3-D AABB measurements are constantly changing as opposed to loads that are lifted in indoor environments and that do not rotate constantly. The correctness or accuracy of the load object measurement cannot be evaluated using this dataset, because it is difficult to interpret as is. If the cylinder fitting is used and the cylinder length is computed using Algorithm 3, it is much easier to see how the normal work cycle affects the measurement values with the same object.

6.5 Results: Bounding Volume Measurement

The results of the bounding volume measurement are difficult to evaluate, because the measurement is taken along the axis-aligned bounding box(AABB). For example, if the camera optical axis is installed at an angle with the coordinate axes, the 3-D AABB bounding volume will contain additional empty space as a function of the angle.

The load object that was used for measurement purposes in datasets A1-A6 and B1-B7 was measured along its minimal bounding volume. The measured load object is found in datasets A and B, which are the indoor test locations. Only the bounding volumes provided by the top-down camera viewpoint A produced meaningful results, because little additional space introduced by the AABB bounding volume was present. In the installation location B, the camera viewpoint B tilts a lot, thus, the bounding volume contains a lot of empty space and the output measurements are not descriptive of the actual dimensions of the object as is. Also the lifting mechanism is part of the load object and cannot be removed properly in viewpoint B. The results of the top-down viewpoint A will be mainly presented.

Detailed analysis was done for datasets A1 and A6 where the top-down viewpoint is in use. Dataset A3 could have also been used for detailed analysis since the load object is being measured in the frame range [200 600] and almost 400 consecutive successful measurements are found as seen in Figure A.3 in Appendix A.

6.5.1 Results: Dataset A1

Figures 6.9 and 6.10 show a single frame from dataset A1 where a load object point cloud and its AABB are seen from the viewpoint A. Both images show a different angle of the scene. In the scene, there was a load object that was not being lifted, and whose dimensions had to be measured. The load object was box-shaped with a wider lower part, and a narrower upper part as seen in Figure 6.10. The load object stayed in the same pose throughout the dataset A. The perception platform was moving, which introduces some additional depth estimation error in the dataset.

The measurement results of the AABB bounding volume from dataset A1 are visualised in Figure 6.7. Full visualisation of the dataset may be seen in Figure A.1

Figure 6.7: Load object AABB measurement results computed from dataset A1 for all frames.

in Appendix A. More importantly, a subset of dataset A1 in the frame range [22 60] was selected for error review of the load object measurement as seen in Figure 6.8. In dataset A1, the load object is rotated and not axis-aligned as can be seen in Figure 6.9. The load object expected value must be estimated to be able to compare the results with the actual object dimensions.

In Figure 6.9 the point cloud and its AABB are visualised as seen from a virtual camera. The actual load object measurements are shown in the picture as X_{meas} and Y_{meas}. The load object measurement software reports the AABB dimensions, which are shown as X_{aabb} and Y_{aabb}. As can be seen, reported measurement X_{aabb} is a much higher value than value X_{meas} in this orientation of the load object. Similarly, the reported measurement Y_{aabb} is also much higher than the value Y_{meas}. The outputs X_{aabb} and Y_{aabb} are formulated as non-linear functions

$$X_{aabb} = max(P(X)) - min(P(X)) \tag{6.1}$$

$$Y_{aabb} = max(P(Y)) - min(P(Y)) \tag{6.2}$$

where P is the point cloud containing all points in the load object point re-

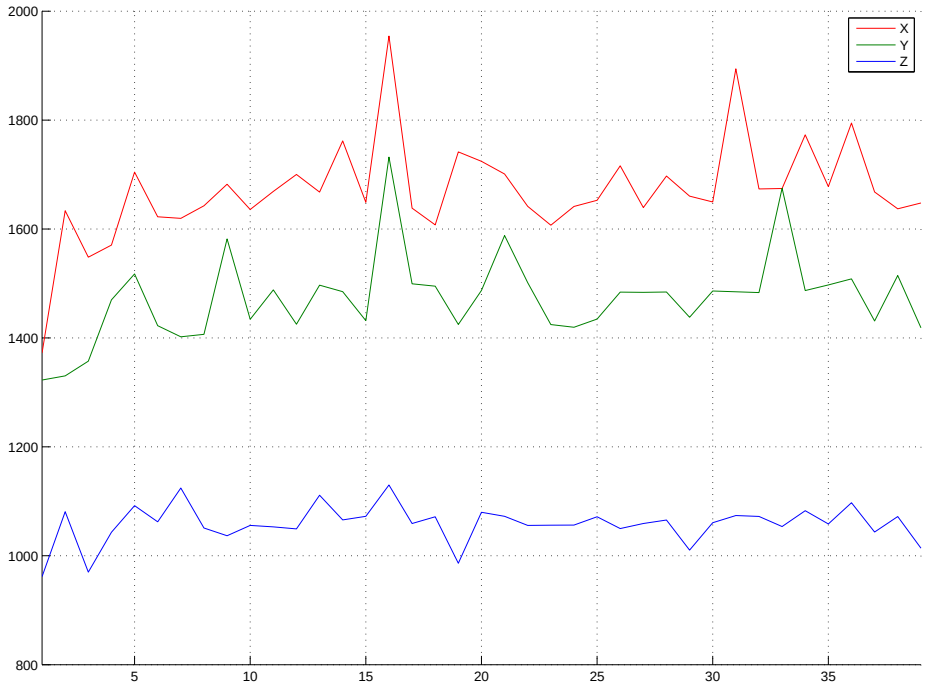

Figure 6.8: Load object AABB measurement results computed from dataset A1 in the frame range [22 60].

sponse. It would be beneficial to find another pair of functions that report the actual load dimensions as a function of the output of the measurement software's AABB dimensions and the orientation of the load object e.g.

$$X_{meas} = f(X_{aabb}, \gamma) \tag{6.3}$$

and similarly for y-direction

$$Y_{meas} = f(Y_{aabb}, \gamma) \tag{6.4}$$

where γ is the measured angle between the load object main axis and the camera frame x-axis. Functions (6.3) and (6.4) are difficult to estimate since they are non-linear, but a simple approximation can be done using trigonometry according to Figure 6.9:

$$X_{meas} \approx t \cos \alpha \tag{6.5}$$

where t is found according to Pythagoras equation

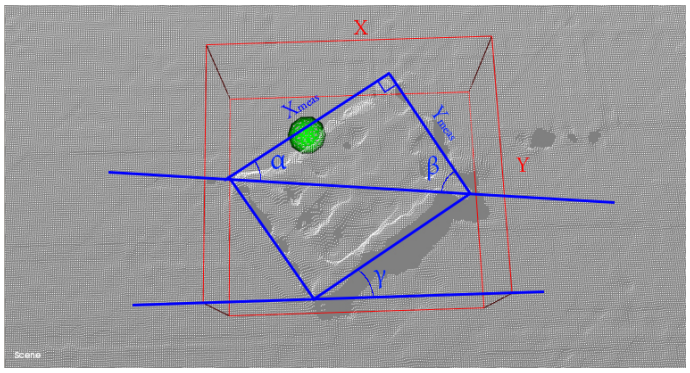

Figure 6.9: The load object point cloud response and its computed AABB bounding volume seen from the top in dataset A1. X_{aabb} is the x-component of the bounding volume along the camera frame x-axis. Y_{aabb} is the y-component of the bounding volume along the camera frame y-axis. The dimensions X_{aabb} and Y_{aabb} are a non-linear function of the rotation of the load object in the world frame.

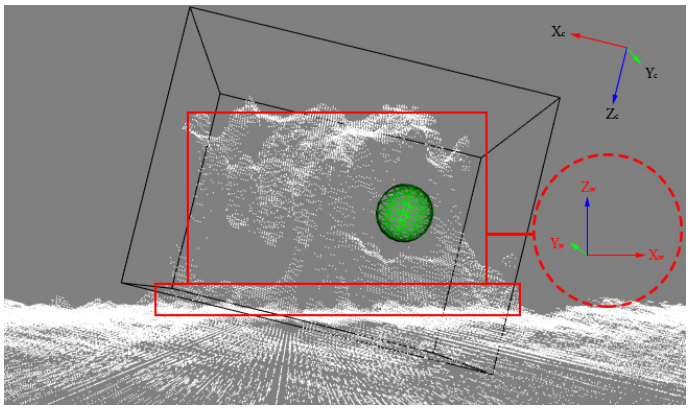

Figure 6.10: The load object point cloud response and its computed AABB bounding volume seen from the side in dataset A1. The data points are not transformed to the world coordinate frame, which results in a tilted AABB bounding volume(in black). The load object actual shape is shown in red.

$$t \equiv \sqrt[2]{X_{meas}^2 + Y_{meas}^2} \tag{6.6}$$

In Figure 6.9 the load object rotation about z axis is $\gamma = 34°$(estimated from the image). Angles α and β are computed using trigonometry

$$\alpha = 31.7361°$$

$$\beta = 58.2639°$$

Table 6.4: Load object actual dimensions and expected AABB values from dataset A1 in the frame range [22 60].

Dimension	**Estimated Expected Value**	**Actual Value**	**Diff. %**
X	1787	1520	+17.6%
Y	1600	940	+70.2%
Z	930	920	+1%

and t is solved

$$X_{aabb} \approx t = 1787.2$$

The value of t is used as the *estimated expected value* of X_{aabb}. According to Figure 6.9 the height h of the triangle can be used as a simple estimate of the expected value of Y_{aabb} according to

$$Y_{aabb} \approx 2h$$

Using trigonometry h will solve as

$$h = Y_{meas} \sin \beta$$

and its solution is

$$h = 799.4511 \approx 800.0$$

Thus, the expected value of Y is 1600. All the actual dimensions and estimated expected values of the AABB dimension measurements are listed in Table 6.4. Estimation of the expected value of Z_{aabb} is the most difficult value to estimate since X_{meas} and Y_{meas} affect Z_{aabb} a lot. Similarly, Z_{meas} does affect X_{aabb} and Y_{aabb} dimensions but the effect is minor since the tilt of the camera frame z-axis is only 6° in Figure 6.10. Taking an error difference from the estimated expected values and the actual measurements in the selected frame range will result in histograms that are shown in Figures 6.11, 6.12, and 6.13. According to the error histograms, the measurement outputs follow normal distribution even though the mean value is shifted from the expected value. This means that the AABB bounding volume estimation of the expected values is not precise. If the mean shift is compensated, then the measurement error can be modeled using normal distribution for each axis. The load object expected value differences are shown in Table 6.5. The difference per cent is calculated as the difference of mean value and the expected value.

Table 6.5: Load object measurement statistics from dataset A1 in the frame range [22 60]. Difference is taken between the estimated expected value and the mean value.

Dimension	**Expected Value**	**Mean**	**Median**	**Std.Dev.**	**Diff. %**
X	1787	1672	1660	91.07	-6.4%
Y	1600	1473	1484	78.02	-7.9%
Z	930	1059	1059	34.72	+13.9%

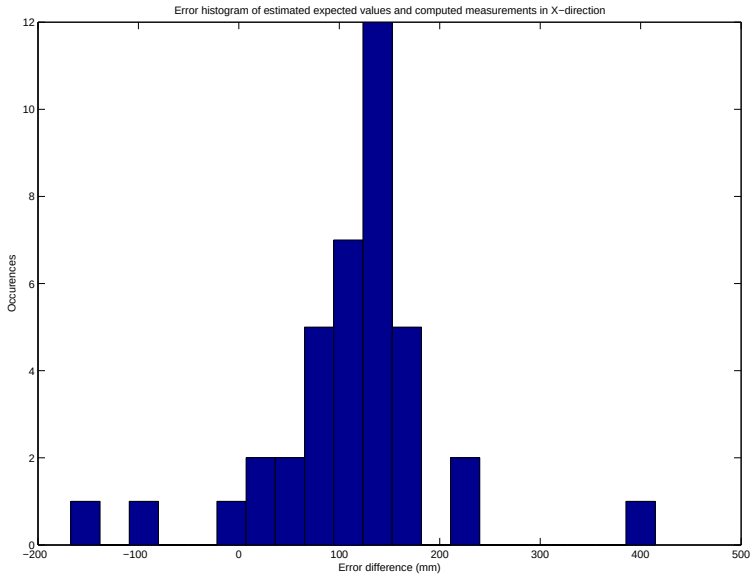

Figure 6.11: Error histogram of the x-direction measurements from dataset A1. The error is computed between an estimate of the expected value (see Table 6.4) and the actual AABB measurement. The mean error of x-direction is 115.4 with a standard deviation of $\sigma_x = 91$.

6.5.2 Results: Dataset A6

Figure 6.14 shows a single frame from dataset A6 where a load object point cloud and its AABB are seen from from a virtual camera in camera installation location A. The load object is being lifted, and the box-shaped object is almost aligned along the camera frame axes. The load object is the same object that was used in dataset A1 and was described previously.

The measurement results of the AABB bounding volume from dataset A6 are visualised in Figure 6.18. Full visualisation of dataset A6 may be seen in Appendix A in Figure A.4. The subset of dataset A6 in the frame range [250 340](see Figure 6.19) was selected to review the error in the load object measurements. The expected estimated values of the AABB dimensions are not calculated since the load object is almost aligned with the camera frame axes. The results can be compared to the actual dimensions straightforwardly here. There will be some error introduced, but the effect of the small γ angle is minor, and it is changing as the load object is being slightly rotated whilst being moved. The load object dimension measurement results are found in Table 6.6.

Taking an error difference from the computed values and the actual values in the selected frame range will result in error histograms that are shown in Figures 6.15,

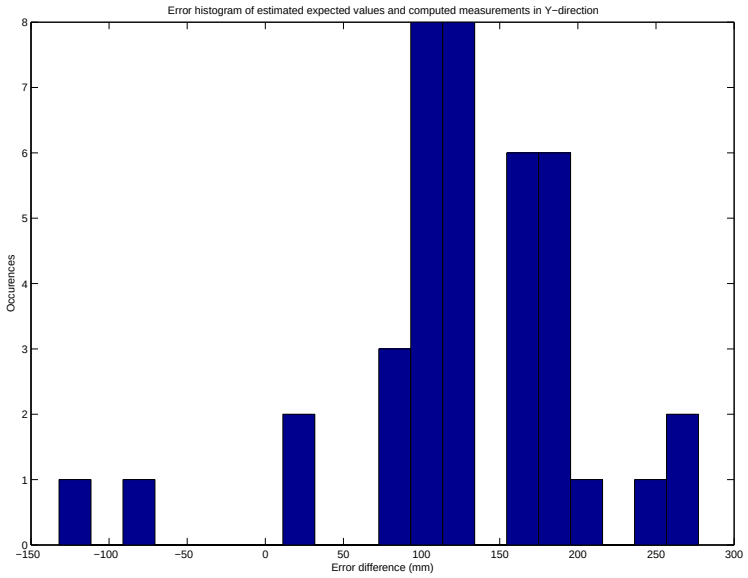

Figure 6.12: Error histogram of the y-direction measurements from dataset A1. The error is computed between an estimate of the expected value (see Table 6.4) and the actual AABB measurement. The mean error of y-direction is 126.8 with a standard deviation of $\sigma_y = 78$.

Table 6.6: Load object measurement statistics from dataset A6 in frame range [250 340]. Difference is taken between the expected value and the mean value.

Dimension	**Expected Value**	**Mean**	**Median**	**Std. Dev.**	**Diff. %**
X	1520	1634	1627	79.74	+7.5%
Y	940	1007	974	92.85	+7.1%
Z	920	910	919	56.11	+1.1%

6.16, and 6.17. According to the histograms, the lifted load object error distributes poorly with normal distribution in this frame range, which means that there is additional noise caused by rotation, changing lighting, moving load object or other disturbances.

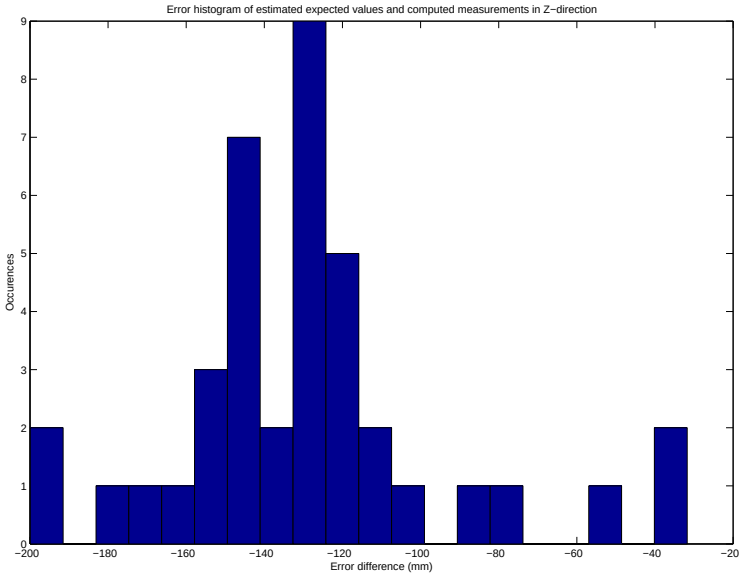

Figure 6.13: Error histogram of the z-direction measurements from dataset A1. The error is computed between an estimate of the expected value (see Table 6.4) and the actual AABB measurement. The mean error of z-direction is -128.5 with a standard deviation of $\sigma_y = 34.7$.

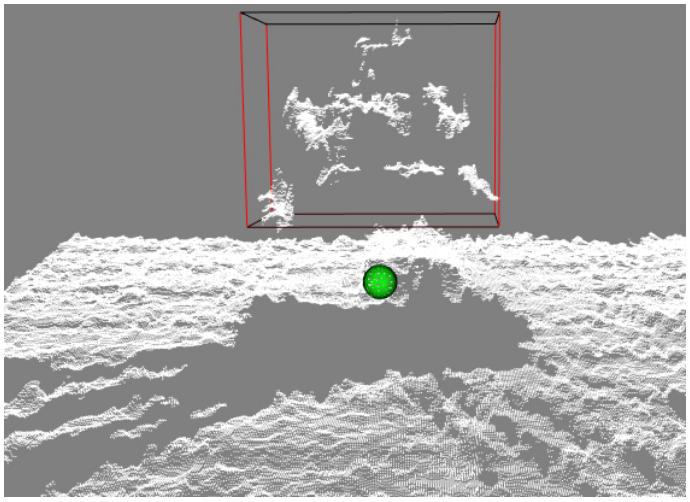

Figure 6.14: The load object is being lifted in dataset A6. The load object has protruding elements on the bottom, which makes the bottom point response very clear and enables a correct bounding volume detection. If a standard shape box is used, the point response on the occluded side of the load may not be good enough to generate a correct measurement using a single camera viewpoint.

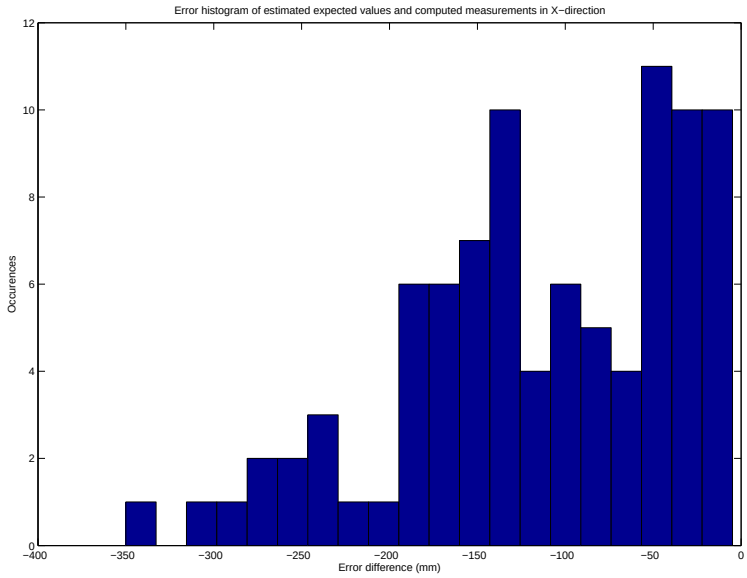

Figure 6.15: Error histogram of the x-direction measurements from dataset A6. The mean error of x-direction is -114.4 with a standard deviation of $\sigma_x = 79.7$.

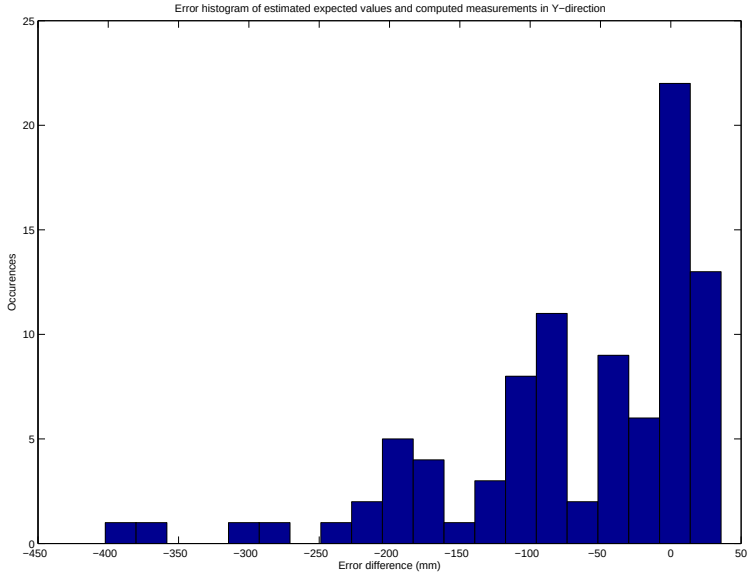

Figure 6.16: Error histogram of the y-direction measurements from dataset A6. The mean error of y-direction is -67.4 with a standard deviation of $\sigma_y = 92.6$.

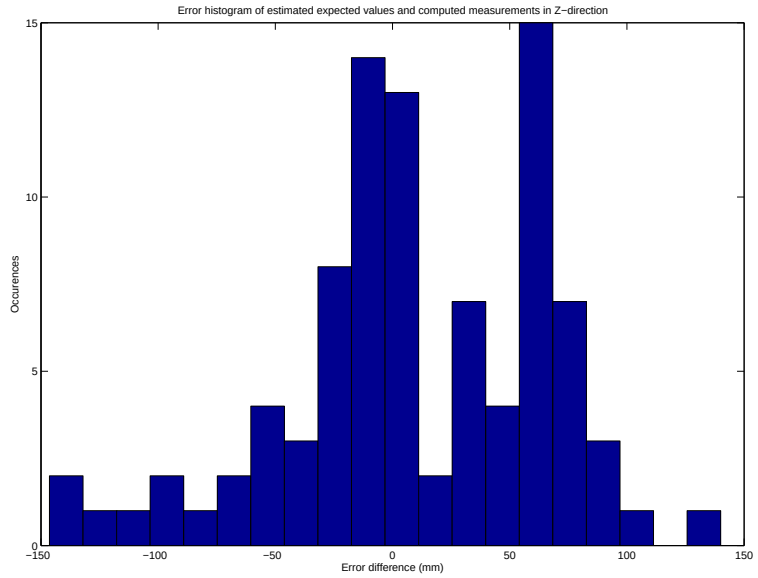

Figure 6.17: Error histogram of z-direction measurements from dataset A6. The mean error of z-direction is 9.8 with a standard deviation of $\sigma_z = 56.1$.

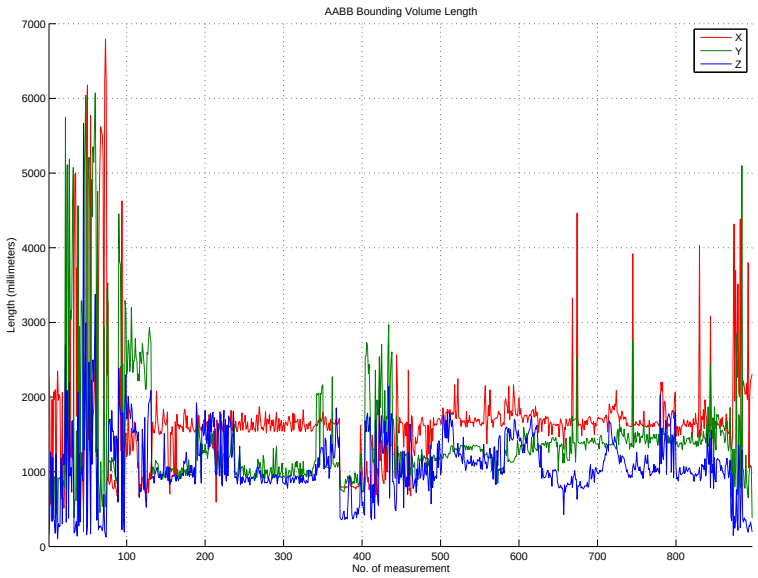

Figure 6.18: Load object AABB measurement results computed from dataset A6 for all frames.

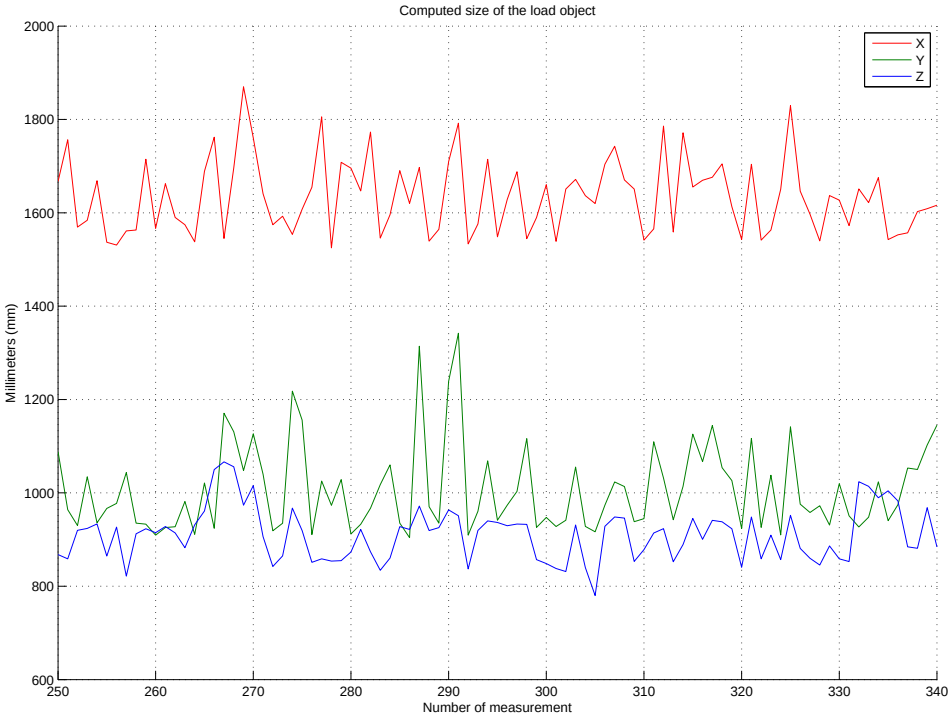

Figure 6.19: Load object AABB measurement results computed from dataset A6 in the frame range [250 340].

Chapter 7

Discussion And Conclusion

7.1 Discussion On Indoor Load Object Measurement

In the indoor case, the performance of the load object measurement software was tested extensively using two selected viewpoints: the top-down viewpoint A and the bird's eye viewpoint B. On the basis of all findings in Chapter 6 the top-down viewpoint A performed faster, and resulted in a successful load object measurement more often than the bird's eye viewpoint. With the bird's eye viewpoint, the load object measurement was especially difficult for far-away object locations near the wall. In light of all findings and analysis the top-down camera configuration A is recommended for load object measurement purposes in the author's opinion.

7.1.1 Effects Of Selected Viewpoints

The output AABB bounding volume from a top-down camera was usable as is, and only a little extra volume was induced by a 6° tilt of the camera frame in tests. In general, the load object measurement output was consistent with the top-down camera installation location A. The ground plane was not always fully modeled due to the moving cameras, occlusion situations, and changing lighting conditions, but the measurement was not affected at all by such modeling problems.

The bird's eye view suffered from increased distance to objects near the wall causing smaller point responses for the load object(due to effectively smaller image area). Additionally, the bird's eye view suffered from merging of objects into the point response of the wall. In dataset B3 the load object is lifted in the air and out of picture near the wall first, and then brought in the near vicinity of the cameras and lifted in the air again. Figure A.5 in Appendix A shows the results, and it can be seen that the AABB performance is poor in the frame range [100 300] where the object is found far from the cameras, but gets better in the range [300 500] when

the load object is nearby. Also, it can be seen from the sensor frame max and min coordinate data that as the load object is being lifted, the z-axis coordinates and y-axis coordinates change according to the lift operation, but the AABB dimensions stay constant as supposed. There is noise in all the measurements, and the nearer the load is being lifted to the cameras, the smaller the noise becomes(see frame range [420 450] that shows minimum noise in the AABB measurement in Figure A.5 in Appendix A). The changing y-coordinate is caused by the tilt of the camera platform, thus, the lift is not solely in the direction of the z-axis. Additionally, the changing perspective projection of the load during the lifting operation does have an effect. In general, it is desirable to keep only a single axis parameter changing during a lift operation, and it is desirable that there is no change in the 3-D AABB output. The single axis parameter changes only if the load object is oriented along the sensor frame axes or transformed into a world coordinate frame.

A lesser problem with the bird's eye viewpoint was the possibility of lifting the load object out of picture. In comparison, it was not possible to lift the load object out of the picture in a top-down camera configuration. Also by using the viewpoint B the hoisting mechanism was modeled as part of the load object. The effect of the hoisting mechanism modeling can be seen in Figure A.6 in Appendix A in the frame range [1200 1500]. The AABB dimensions of the load change as the load is being lifted, which is an unwanted phenomenon. The hoisting mechanism should be removed from the load object point cloud to prevent a change in the AABB constants during a lift operation. Also, the load object should not be lifted out of the camera image.

7.1.2 Load Object Measurement Results

The measurement values produced by the software could be compared with the actual dimensions only after computing the estimated expected values. It is apparent that a single measurement will not report the correct bounding volume, but it is a better idea to report the median or average bounding volume of a series of e.g. 10-15 measurements. Since the *Himmeli* platform can successfully deliver 4 frames per second without skipping frames, a load object measurement would currently take 3-5 seconds of time. The decision of how to interpret the constant flow of load object measurement data in the ROS network is left as the responsibility of the software that subscribes to the provided data feed. The accuracy of the measurement depends on many factors and with the offline testing a maximum average error of $< 15\%$ was reached. Currently, the tests showed an average maximum of $+13.9\%$ difference in the dimensions measurement, and the range was within $[-7.9\ 13.9]$ in percentage for datasets A1 and A6.

The accuracy of the measurement is highly dependent on the amount of noise in the data, and accuracy of the sensor that generates the point cloud data. If the

data quality cannot be improved from the sensor side, then signal processing filters such as a Kalman filter should be constructed to filter the bounding volume output over time in order to better identify the noise and the signal parts.

7.2 Discussion On Outdoor Load Object Measurement

In the outdoor case, the performance of the load object measurement system was tested using two viewpoints: the frog perspective viewpoint C and the bird's eye view (II) D. On the basis of all findings in Chapter 6, the bird's eye viewpoint may be better for measuring load objects in outdoor applications. The frog perspective viewpoint does not cover the sides of the working area effectively, and the load object can be moved out of the picture. Also the bird's eye viewpoint has its flaws: a boom that occludes the load object during normal operation is visible in the camera image. The bird's eye viewpoint has one major advantage over the frog perspective: if the stereo cameras are mechanically attached to an outdoor work machine booth that is vibration free, the image will be much more stabilised than in any other installation.

7.2.1 Load Object Measurement Results

Load object measurement used simple cylinder segmentation in the outdoor tests, and the resulting cylinder extraction result can be seen in Figure 7.1. There are no exact statistics on the extracted load orientation error since cylinder fitting was not a researched target in this work. Better heuristic checks for the correct load object selection are suggested as an improvement to the cylinder segmentation. For example, it is possible to select a wrong cylindrical load object from data points located below the end effector. Two simple conditions that would improve the current cylinder selection significantly would be

- to check that the shortest distance from the end effector to the closest point along the cylinder main axis does not exceed a pre-defined threshold
- and to check that the distance between the centroid of the segmented cylinder and the end effector does not exceed a pre-defined threshold.

For example, if the distance between the candidate cylinder and the end effector is too large, then the candidate should be rejected. Improvements in the load selection algorithm are easy to implement, but due to limited development time the conditions were not further advanced.

In Figure 7.1, the stereo parameters and the object segmentation parameters were not optimally tuned. Still, the correct cylinder extraction was quite robust in the presence of missing data, noisy 3-D data, and non-optimal parameter tuning.

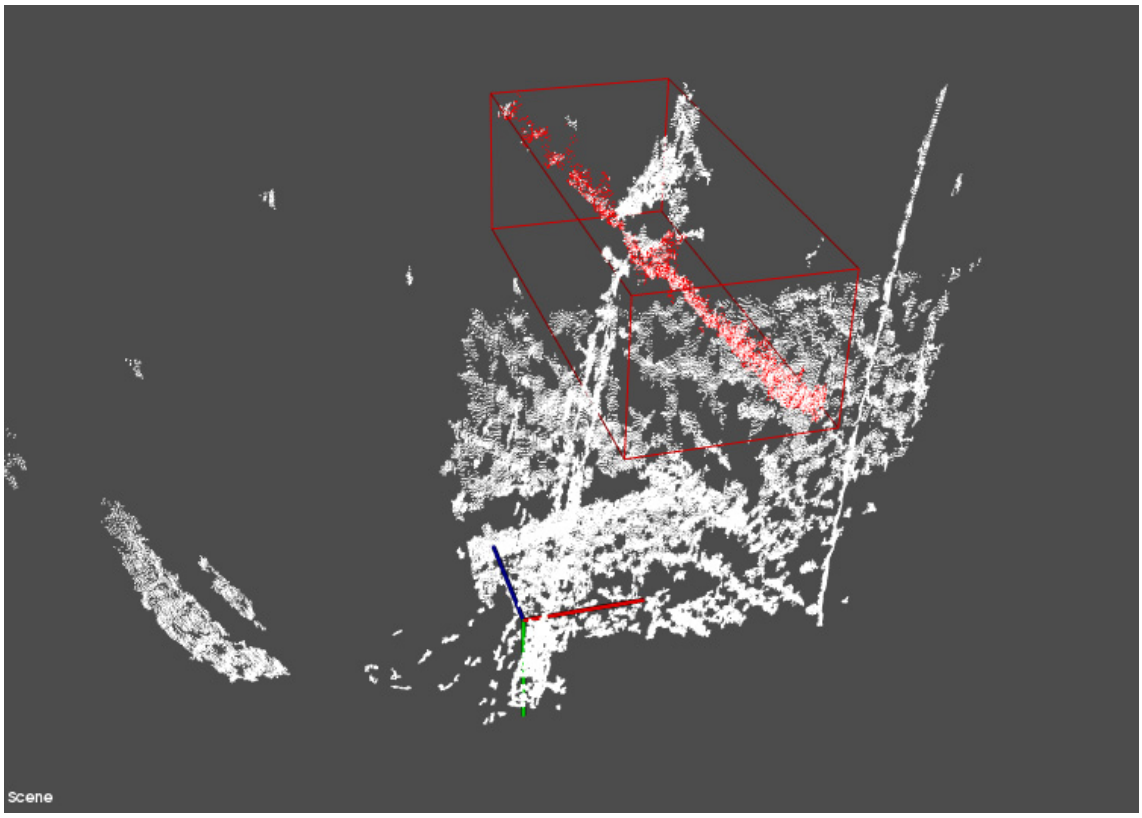

Figure 7.1: Load object detected. The data is block matched using the quick BM algorithm and the point cloud contains a low number of points for fast 3-D computation times. The missing data on the left side of the ground plane is caused by occlusion(by the lifting mechanism).

The most significant problem in the load object measurement was the fusing of multiple objects in the load object point cloud. While the cylinder extraction does well on selecting and extracting the correct feature if it remains alone, it is unknown whether the object segmentation step can really handle situations where multiple objects are grouped as a single large point cloud.

A minor problem with measuring loads was introduced when the bird's eye view (II) was used and the loads were occluded by a boom. An object may be seen on both sides of the boom, which means that in the 3-D data two point clouds are seen that represent partial surfaces of the load object. In these cases where the load object was occluded by the boom and divided into multiple point clouds, it was not possible to measure the load object size correctly.

7.3 Discussion On Bottlenecks In The Measurement System Operation

Since the used stereo cameras can deliver images at a steady rate of up to 50 FPS, the perception platform software running on Windows was one of the major bottlenecks with its capability of transmitting stereo images only at a rate of 4 FPS through the network. Another bottleneck was detected in the client software visualisation updating, which randomly took a lot of time. On average, a random slowdown would last almost a full second before a single measurement was finished. In Appendix A, the spikes in the computation time charts show the slowdowns in detail, and the highest spike can be seen in Figures A.6 and A.8 lasting as long as 4000 milliseconds. For most measurements, the time needed for processing 3-D data into measurement information was 100 milliseconds per measurement on average.

Computation time could be reduced by removing visualisations from the load object measurement client. Also, the C++ implementation of the software could support more parallel programming and GPU implementations for select point cloud operations. Finally, lower resolution cameras could be used, or the image ROI could be limited to only contain the image of the load object to reduce the amount of data to effectively speed up the computation.

7.4 Discussion On Point Cloud Geometry Scale Verification

3-D data geometry that was output from the stereo camera process was assumed to be geometrically correct when a calibrated stereo system was used. This assumption may or may not be true, which is why the 3-D data geometries of the measured load objects were studied. The verification was done simply by comparing the dimension measurements produced by the load object measurement software and the actual measures of the load object. Unfortunately, the geometry correctness of the point cloud cannot be globally verified using only local 3-D feature results. Since the dataset environments did not contain any installed landmarks, a scale verification was not available.

For position accuracy verification and point cloud scale verification purposes, landmark objects could be installed in the environment, and their correct locations could be searched for and verified from the point cloud data. A suggested improvement would be to install 3-D spheres as control points in the test environment prior to dataset capture. Then, the reconstructed geometry can be verified by comparing the computed locations of the spheres with their actual locations. If the 3-D data locations of the control points do not match their actual locations, an unsuccessful

camera calibration or a scale difference can be easily detected.

7.5 Discussion On Perception Platform

Using the *Himmeli* perception platform as the medium for stereo camera processing was a successful decision. The platform provided ROS and PCL computing hardware at the ready, and the installed perception sensors were of high quality. In bright daylight, the camera image was good.

The sturdy metal frame protected the stereo cameras from moving or dislocating during platform installations or transport from test site to another, and the modularity was useful in installing the cameras in tight spaces. The *Himmeli* platform was heavy, which helped passively stabilise the camera images in vibrating environments, but made moving the platform a bit difficult. Using the stereo cameras installed as part of the *Himmeli* perception platform may improve the image quality due to passive damping properties of a larger mass of the platform body.

The platform handled temperatures below zero well, and the system could be operated even in winter conditions if the camera lens casings are installed with heaters and a solution that keeps the optics clear of water, dirt, and snow.

The stereo camera image capture was limited at 4 FPS, because a higher rate of image capture, even 5 FPS resulted in skipping of frames, which lead to a situation where many frames with different timestamps had the same image content. If a dataset includes such inconsistent content in frame pairs, the stereo calibration fails, and stereo reconstruction assumptions do not hold any more.

The platform included a LIDAR scanner that was used to collect data for object segmentation testing, which was an invaluable asset in the work. The Velodyne HDL-32E scanner was also a secondary perception sensor that could have been used if the stereo camera setup would have proved to be not suitable for load object measuring use.

Finally, the perception platform hardware suffered from minor problems, such as one of the PC computers halted during heavy processor loads due to inadequate dissipation of heat from the processor. Another problem was encountered when the stereo cameras were initialised, because of problems in left and right camera assigning order.

7.6 Discussion On Prototype Software

A lot of time was used in the development of the load object measurement software. The software structure followed a standard ROS package implementation, and most of the required computation algorithms were readily available as part of the *OpenCV* and the PCL software libraries. The software package ran some advanced algorithms,

such as RANSAC computation and k-nearest neighbour search, but they were not visible for the user. It would have been possible to implement some portions of the software better than what is currently implemented, for example, an end effector tracker in the indoor case would have been useful. Due to limited resources most computations use readily available external library functions, and only some heuristics and algorithms were implemented by the author. In general, the software is usable with single button solutions for logging data from new image datasets. If the camera platform is readily calibrated, the user only needs to configure new launch files that go by a unique dataset title, and choose the parameters accordingly.

The PCL library handled uninitialised data well, which resulted in robust functionality of the client software. Moreover, the software architecture did not depend on sensor frame orientation information, which is why the computation works independent of the perception platform orientation.

The ROS network that runs the load object measurement system can be difficult to configure without good knowledge of the ROS system and the Linux operating system. To make the usage of the software easier, launch files were written for offline and online testing purposes, and they accelerated the testing a lot by e.g. launching multiple ROS nodes simultaneously.

A software version was branched out that was used to log the load object measurement data in an XML file format for processing into visuals that are collected in Appendix A. Visualisations of the data proved to be invaluable help in evaluating whether the load object measurement system worked correctly.

Finally, the development of the prototype had issues with standard development environments, such as Eclipse, because multiple external libraries were being integrated in the work. In the end, the C++ code was compiled using purely *cmake*.

Discussion On Voxel Grid Filtering

The incoming point cloud was downsampled using the PCL voxel grid filter so that the number of output points stayed in a range of 2-20% of the original number of valid data points. The range was found out by testing different values of the voxel grid leaf size parameter that satisfied a goodness criterion.

The criterion used for selecting the voxel grid leaf size was the output length measurement for the load object in x-, y- and z-directions with error limits. If all length measurements stayed within reasonable error limits, for example, 3-5% error compared to the measurement from the original point cloud, while the number of 3-D data points was reduced, the downsampling would meet the criterion. Besides the error limits, two constraints were used in the evaluation of the grid filter leaf size tuning: the 3-D features must not split in the nearest neighbour search, and small details must retain a volume.

The first additional constraint translates to leaf size being smaller than the near-

est neighbour search radius parameter used in the segmentation search. Since this constraint has an effect on the criterion, a visual inspection was required to make sure that the correct 3-D feature was used in the criterion.

The second additional constraint was enforced visually so that no features were reduced into less than 4 data point entries. A minimum of 4 data points are required to span a 3-D volume.

A value of 120 millimeters was selected as the voxel cube leaf size, which produced a point cloud downsampled down to a 2.2% size of the original point cloud. This value was used throughout the computation of the results presented. The value was found using a parameter tuning tool that was programmed for the task. In the software, a leaf size parameter slider changed the cube size, and the effects could be instantly visualised.

Discussion On Ground Plane Removal Using RANSAC Estimator

In the implemented software, the RANSAC algorithm found the ground plane successfully for almost all frames and removed the ground well, which means that the selected error threshold of 14 cm was a good value. The number of successful ground removals was not tracked and is not known. The ground plane removal was robust against missing parts of the floor data and occlusions in testing.

There were some conditions that were fulfilled in order to find the correct solution for ground plane removal. First, the ground plane must contain the largest number of points in a planar manner found in the dataset. If not, then another plane will be found by the algorithm, for example, a large warehouse wall can be found in the RANSAC search.

Secondly, the plane response must be within the selected error threshold. In the bird's eye view some points were not removed that fell outside the error threshold due to high depth estimation error and curving of the floor-wall interface(described in Section 6.4.1). Similarly, the ground plane estimator will fail to remove grounds that are not planar, e.g. rolling hills or mounds of dirt in the ground.

Discussion On Cylinder Model Estimation

Some problems in the cylinder segmentation were encountered when the load object was lifted using a boom. The estimator easily fitted the cylinder model on the point cloud representation of the boom and not on the cylindrical load object, which was an incorrect result. To prevent selection of wrong data, a set of simple heuristic checks was implemented in the software. The results of the cylinder model estimation were good since the estimator found the correct cylindrical load object from most frames when cylindrical loads were lifted in the air. The number of correct detections of cylindrical objects was not tracked.

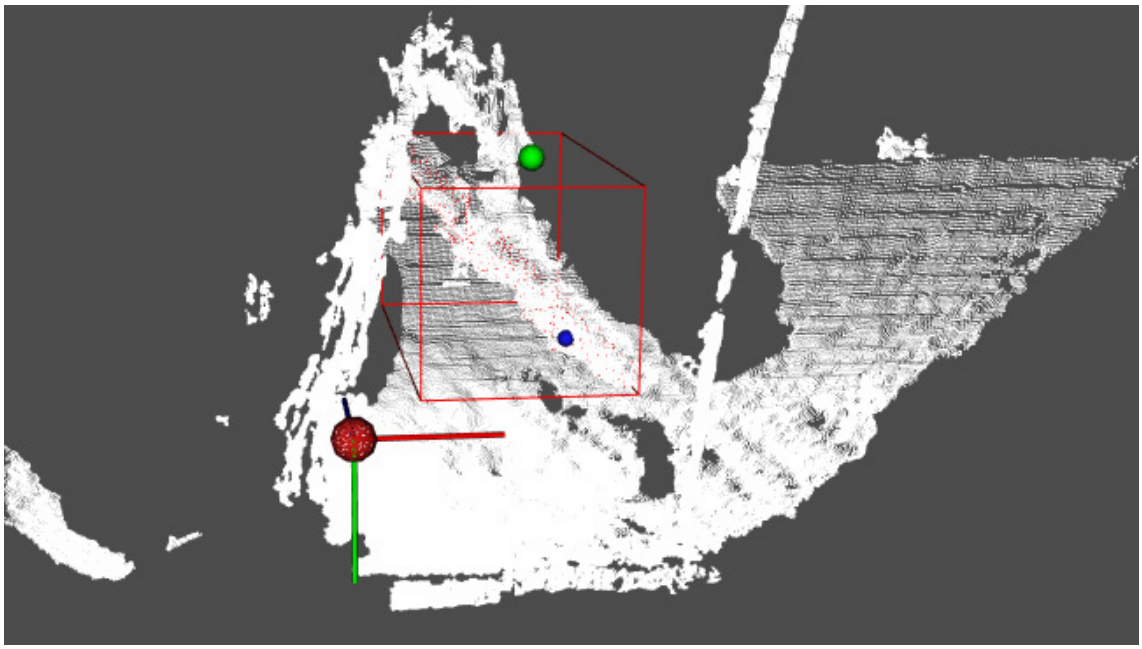

Figure 7.2: Bounding volume computed from cylinder model segmentation result shown on top of original point cloud data.

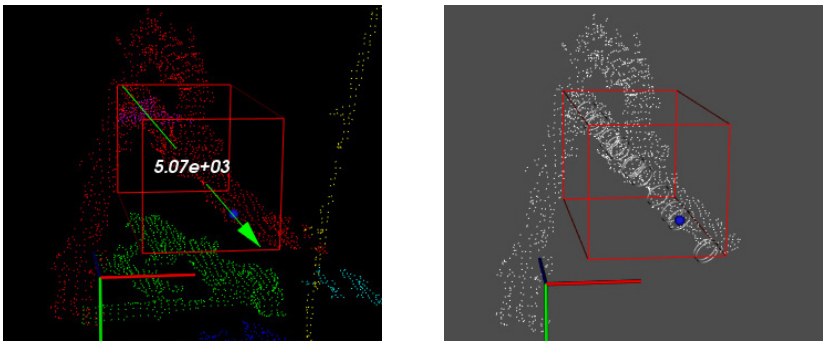

Figure 7.3: a) Cylinder length in millimeters in the segmented object view b) Cylinder visualisation in the load object point cloud.

The cylinder segmentation could return a correct result for a load object that was on the ground merged with other objects if the data was not too cluttered. A correct detection happened approximately for half of the frames, but there was large errors in the cylinder orientation. A typical dataset that was tested contained 200-1600 stereo image pairs.

In general, the radius of the cylindrical load object was estimated slightly too large, which was acceptable in a bounding volume computation. There are no statistics on the average cylinder radius estimation results. The cylinder model segmentation was moderately fast with the test data: less than 10 milliseconds of time was used on cylinder segmentation computation on average.

A cylinder segmentation result can be seen in Figure 7.2 where the inliers of the downsampled cylinder model are shown as red points, and the bounding volume of the cylinder load object is shown as a red cuboid. The length of the segmented cylinder can be seen in Figure 7.3a in millimeters(5007 mm). The cylinder visualisation is seen in Figure 7.3b, which contains only the load object point cloud data. It can be seen that the load object point cloud contains the boom, and there is some orientation and translation error in the found cylinder parameters. The actual length of the lifted cylindrical load was not measured.

7.6.1 Discussion On Bounding Volume Computation

The 3-D OBB could be easily implemented if eigenvectors of the load object point cloud are computed. A 3-D OBB will report the actual bounding volume more accurately than the 3-D AABB bounding volume, and it is invariant to rotations of the load object in the world coordinate frame. The OBB perimeter can be oriented along the largest eigenvector in a right-hand normalised coordinate system. Orienting the main axis along the largest eigenvector does not optimally minimise the bounding volume, but it does make the bounding volume more invariant to changes in the load object orientation. The 3-D OBB can be computed with the PCL *Moment of inertia and eccentricity based descriptors* -module, which is supported by PCL versions 1.7 and higher. The OBB computation was not implemented, because ROS Groovy does not support PCL 1.7.

7.7 Conclusion

A three-dimensional measurement of the load object is required to enable new automated features in traditionally manual crane environments. The crane environment is dynamic and people may work in the area. For example, collision avoidance technique is a new feature for manual cranes that can be realised by using a three-dimensional load object measurement.

In this work, the 3-D measurement of the load object was based on passive stereographic triangulation and 3-D reconstruction. The visible surface model of the crane environment was computed with a calibrated machine vision system. The test environments were actual, non-controlled, dynamic industrial environments that were challenging to reconstruct in 3-D. The bounding volume of the load object was successfully computed from the visible surface data, which meant that the goal of this work was met.

The parameters tuned in the stereo processing were the best possible trade-off between accuracy, quality, and number of points in the resulting point clouds. The number of points in a point cloud heavily affected the time used for measurement computation time. The computation time of a single load object measurement was on average 100 milliseconds. The camera hardware limited the speed to 4 measurements per second for practical reasons.

In indoor locations, an accuracy of $\pm 15\%$ of the actual dimension value on average was achieved. Camera viewpoint above the load object generated the best results. A 2-D image tracking of the load object or a load object 3-D position signal was required to be able to select the correct load object.

In outdoor locations, the accuracy of the measurement was not analysed. The 3-D reconstruction was more challenging, because the lighting conditions were dynamically changing, and the the load objects were occluded by the lifting mechanism. In the outdoor application, cylinder fitting was used to report the dimensions of cylindrical load objects in better accuracy. Missing data from the back side of the load object could not be compensated with a cylinder fitting, but the orientation and length of the loads were recovered well.

For each dataset that was recorded, a set of parameters was tuned by visual inspection of the output point cloud quality. Robot Operating System was used to launch a time series image data feed that was computed into a 3-D bounding volume coordinates and logged in a time series. Only offline datasets were used, and no online measurements were available in this work.

The accuracy of the developed measurement system was not high, but the system was a low-cost solution suitable for measuring load dimensions in an industrial environment. Currently, the solution is not suitable for accurate positioning or high accuracy measurements applications, which prevents the use in safety applications. Additional development is required to improve accuracy and reliability.

Bibliography

[1] N. Zrni and K. Hoffmann. Development of Design of Ship-To-Shore Container Cranes: 1959–2004. *International Symposium on History of Machines and Mechanisms*, pages 229–242, 2004. DOI 10.1007/1-4020-2204-2-19.

[2] K. Rintanen. Modeling, Instrumentation and Modeling Of A Trolley Crane. *Report 87*, 1991. Säätötekniikan laboratorio, Helsinki University of Technology, http://autsys.aalto.fi/attach/Julkaisut/r87isbn9512208318.pdf. Accessed on June 23rd 2014.

[3] H. Laitasalmi. Konenäköön perustuva nosturin käytön ja turvallisuuden seuranta, 2013. Master's thesis. Aalto university School of Electrical Engineering (ELEC).

[4] B. Leibe, K. Schindler, and L. Van Gool. Coupled Detection And Trajectory Estimation For Multi-object Tracking. In *11th Int. Conf. on Computer Vision (ICCV)*, pages 1–8, 2007. DOI 10.1109/ICCV.2007.4408936.

[5] C. Miller, B. Allik, M. Ilg, and R. Zurakowski. Kalman Filter-based Tracking Of Multiple Similar Objects From A Moving Camera Platform. In *51st Annual Conf. on Decision and Control (CDC)*, pages 5679–5684, 2012. DOI 10.1109/CDC.2012.6425956.

[6] L. Lun-Hui, H. Chung-Hao, and Y. Zhi-Heng. Efficient Visual Feedback Method To Control A Three-dimensional Overhead Crane. In *IEEE Transactions on Industrial Electronics*, volume 61, pages 4073–4083, 2014. DOI 10.1109/TIE.2013.2286565.

[7] A. Kaneshige, T. Akamatsu, and K. Terashima. The Development Of An Autonomous Overhead Traveling Crane With Real-time Path Planning Based On The Potential Method. In *Int. Conf. on Control and Automation (ICCA)*, volume 2, pages 1079–1084, 2005. DOI 10.1109/ICCA.2005.1528282.

[8] K. Peng and W. Singhose. Crane Control Using Machine Vision And Wand Following. In *Proceedings of IEEE Int. Conf. on Mechatronics*, pages 1–6, 2009. DOI 10.1109/ICMECH.2009.4957227.

[9] K. Peng, W. Singhose, and P. Bhaumik. Using Machine Vision And Hand-motion Control To Improve Crane Operator Performance. In *IEEE Transactions on Systems, Man and Cybernetics*, volume 42, pages 1496–1503, 2012. DOI 10.1109/TSMCA.2012.2199301.

[10] A. Rahmani, J. Haddadnia, A. Sanai, and O. Seryasat. Intelligent Detection Of Electrical Equipment Faults In The Overhead Substations Based Machine Vision. volume 2, pages 141–144, 2010. DOI 10.1109/ICMEE.2010.5558471.

[11] S. Nagai, A. Kaneshige, and S. Ueki. Three-dimensional Obstacle Avoidance Online Path-planning Method For Autonomous Mobile Overhead Crane. In *Int. Conf on Mechatronics and Automation (ICMA)*, pages 1497–1502, 2011. DOI 10.1109/ICMA.2011.5985971.

[12] Y. Yoshida and K. Tsuzuki. Visual Tracking And Control Of A Moving Overhead Crane Load. In *IEEE Int. Workshop on Advanced Motion Control*, pages 630–635, 2006. DOI 10.1109/AMC.2006.1631733.

[13] D. Hainsworth, P. Corke, and G. Winstanley. Location Of A Dragline Bucket In Space Using Machine Vision Techniques. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, volume 6, pages 161–164, 1994. DOI 10.1109/ICASSP.1994.389916.

[14] S. Westerberg, I. Manchester, U. Mettin, P. La Hera, and A. Shiriaev. Virtual Environment Teleoperation Of A Hydraulic Forestry Crane. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4049–4054, 2008. DOI 10.1109/ROBOT.2008.4543833.

[15] J. Raula and T. Mustonen. Modelling and Simulation for Safety Design and Assessment. Technical report, Finnish Metals and Engineering Competence Cluster (FIMECC).

[16] S. Fuchs. Calibration and Multipath Mitigation for Increased Accuracy Of Time-of-Flight Camera Measurements in Robotic Applications, 2012. Master's thesis. TU Berlin.

[17] L. You, Y. Ruichek, and C. Cappelle. 3-D Triangulation Based Extrinsic Calibration Between A Stereo Vision System And A LIDAR. In *14th Int. Conf. on Intelligent Transportation Systems (ITSC)*, pages 797–802, 2011. DOI 10.1109/ITSC.2011.6082899.

[18] J. Hjelmstad. 3-D Imaging Using Novel Techniques In Ultra-wideband Radar. In *30th European Microwave Conference*, pages 1–4, 2000. DOI 10.1109/EUMA.2000.338736.

[19] S. Rusinkiewicz and M. Levoy. QSplat, A Multiresolution Point Rendering System For Large Meshes. In *27th Annual Conf. on Computer Graphics and Interactive Technologies (SIGGRAPH)*, pages 343–352, 2000. DOI 10.1145/344779.344940.

[20] pointclouds.org PCD File Format. http://pointclouds.org/documentation/tutorials/pcd_file_format.php. Accessed June 6th 2014.

[21] A. Hornung. Octomap - An Efficient Probabilistic 3-D Mapping Framework Based on Octrees, 2010. http://octomap.github.io/. Accessed July 2nd 2013.

[22] Stanford.edu. The Stanford 3D Scanning Repository, 2013. https://graphics.stanford.edu/data/3Dscanrep/. Accessed July 3rd 2014.

[23] B. Mederos, L. Velho, and L. Figueiredo. Moving Least Squares Multiresolution Surface Approximation, 2003. http://w3.impa.br/~boris/sib2003.html. Accessed July 3rd 2014.

[24] P. Rander, P. Narayanan, and T. Kanade. Virtualized Reality: Constructing Time-Varying Virtual Worlds From Real World Events. In *In proceedings of Visualization '97.*, pages 277–283, 1997. The Robotics Institute Carnegie Mellon University.

[25] pointclouds.org Visualisation Library. PCL Visualization documentation, 2014. http://pointclouds.org/documentation/overview/visualization.php. Accessed July 3rd 2014.

[26] W. Hux. Generating A Mesh, 2007. US Patent 7,173,615. http://www.google.com/patents/US7173615. Google Patents. Accessed September 29th 2014.

[27] R. Chilton, C. Crane, and C. Kuk. Analysis Of Data Structures Used For Storing And Processing 3D LADAR Data. In *Int. Conf. on Control Automation and Systems (ICCAS)*, pages 1496–1501, 2010.

[28] S. Laine and T. Karras. Efficient Sparse Voxel Octrees. In *IEEE Transactions on Visualization and Computer Graphics, Issue 8*, volume 17, pages 1048–1059, 2011. DOI 10.1109/TVCG.2010.240.

[29] R. Shrout. John Carmack on id Tech 6, Ray Tracing, Consoles, Physics and more, 2008. http://www.pcper.com/reviews/Graphics-Cards/John-Carmack-id-Tech-6-Ray-Tracing-Consoles-Physics-and-more?aid=532. Accessed June 14th.

[30] F. Ouyang and T. Zhang. Octree-based Spherical Hierarchical Model For Collision Detection. In *10th Intelligent Control And Automation Congress (WCICA)*, pages 3870–3875, 2012. DOI 10.1109/WCICA.2012.6359118.

[31] H. Noborio, S. Saeki, and T. Ikuta. A Comparative Study Between Real Force Made In Experiment And Virtual Force Made In Octree-based Algorithm. *8th IEEE Int. Workshop on Robot and Human Interaction*, pages 344–350, 1999. DOI 10.1109/ROMAN.1999.900364.

[32] A. Azim and O. Aycard. Detection, Classification And Tracking Of Moving Objects In A 3-D Environment. pages 802–807, 2012. DOI 10.1109/IVS.2012.6232303.

[33] S. El-Hakim, C. Brenner, and G. Roth. A Multi-sensor Approach To Creating Accurate Virtual Environments. In *ISPRS Journal of Photogrammetry and Remote Sensing, Issue 6*, volume 53, pages 379–391, 1998. DOI 10.1016/S0924-2716(98)00021-5.

[34] H. Kawai, Y. Kim, and Y. Choi. Measurement of a Container Crane Spreader Under Bad Weather Conditions by Image Restoration. In *IEEE Transactions on Instrumentation and Measurement*, volume 61, pages 35–42, 2012. DOI 10.1109/TIM.2011.2161830.

[35] Y. Chen and G. Medioni. Object Modeling By Registration Of Multiple Range Images. In *Int. Conf. on Robotics and Automation*, volume 3, pages 2724–2729, 1991. DOI 10.1109/ROBOT.1991.132043.

[36] T. Joshi, N. Ahuja, and J. Ponce. Structure And Motion Estimation From Dynamic Silhouettes Under Perspective Projection. In *Fifth International Conference on Computer Vision*, pages 290–295, 1995. DOI 10.1109/ICCV.1995.466927.

[37] H. Young and R. Freedman. *University Physics with Modern Physics, 13th edition.* 2012. Pearson, ISBN 978-0-321-69686-1.

[38] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision.* 2007. PMF algorithm on page 483.

[39] D. Marr. *Vision.* 1982. W.H. Freeman and Company, ISBN 0-7167-1284-9.

[40] Intel Corporation Russia, Willow Garage, and Itseez. Camera Calibration and 3-D Reconstruction, 2014. http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. Accessed May 23rd 2014.

[41] E. Trucco and A. Verri. *Introductory Techniques For 3-D Computer Vision.* 1998. Prentice Hall, ISBN 0-13-261108-2.

[42] P. Corke. *Robotics, Vision and Control: Fundamental Algorithms In MATLAB.* 2011. Springer, DOI 10.1007/978-3-642-20144-8.

[43] J. Heikkilä. Geometric Camera Calibration Using Circular Control Points. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 22, pages 1066–1077, 2000. DOI 10.1109/34.879788.

[44] Caltech. Camera Calibration Toolbox for MATLAB. 2010. http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html. Accessed June 18th 2013.

[45] J. Heikkila and O. Silvén. A Four-Step Camera Calibration Procedure With Implicit Image Correction. In *Conf. on Computer Vision and Pattern Recognition, IEEE Computer Society*, pages 1106–1112, 1997. DOI 10.1109/CVPR.1997.609468. http://www.vision.caltech.edu/bouguetj/calib_doc/papers/heikkila97.pdf. Accessed May 11th 2014.

[46] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library.* O'Reilly Media, 2008.

[47] S. Mezouari, G. Muyo, and A. Harvey. Circularly Symmetric Phase Filters For Control Of Primary Third-order Aberrations: Coma And Astigmatism. In *The Journal of the Optical Society of America A*, volume 23, pages 1058–1062, 2006. DOI 10.1364/JOSAA.23.001058.

[48] Camera Calibration Toolbox for MATLAB References, 2000. http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/ref.html. Accessed April 8th 2014.

[49] D. Brown. Close-Range Camera Calibration. *Photogrammetic Engineering, No. 8*, 37:855–866, 1971.

[50] J. Craig. *Introduction To Robotics: Mechanics And Control.* 2005. 3rd edition. Appendix B *The 24 angle-set conventions.*

[51] S. Terho. Using Stereo Cameras in MaCI. In *1st GIM International Symposium for GIMnet/MaCI Developers*, 2010.

[52] J. Gluckman and S. Nayar. Rectifying Transformations That Minimize Resampling Effects. In *IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, CVPR.*, volume 1, pages 111–117, 2001. DOI 10.1109/CVPR.2001.990463.

[53] J. Salvi, X. Armangue, and J. Pages. A Survey Addressing The Fundamental Matrix Estimation Problem. In *Int. Conf. on Image Processing*, volume 2, pages 209–212, 2001. DOI 10.1109/ICIP.2001.958461.

[54] A. Carro and J. Morros. An Adaptive Robust Fundamental Matrix Estimation Approach. In *3DTV-CON*, pages 1–4, 2012. DOI 10.1109/3DTV.2012.6365452.

[55] M. Pollefeys, R. Koch, and L. Van Gool. A Simple And Effective Rectification Method For General Motion. In *IEEE Int. Conf. on Computer Vision*, volume 1, pages 496–501, 1999. DOI 10.1109/ICCV.1999.791262.

[56] docs.opencv.org. OpenCV Documentation, Remap function. http://docs.opencv.org/modules/imgproc/doc/geometric_transformations.html Accessed June 30th, 2014.

[57] G. Grevera and J. Udupa. An Objective Comparison of 3-D Image Interpolation Methods. In *IEEE Transactions on Medical Imaging, Issue 4*, volume 17, pages 642–652, 1998. DOI 10.1109/42.730408.

[58] D. Gallup, J-M. Frahm, P. Mordohai, and M. Pollefeys. Variable Baseline/Resolution Stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008. DOI 10.1109/CVPR.2008.4587671.

[59] F. Kooi and A. Toet. Visual Comfort Of Binocular And 3-D Displays. In *Displays*, volume 25, pages 99–108, 2004. DOI 10.1016/j.displa.2004.07.004.

[60] W. Kim, A. Ansar, R. Steele, and R. Steinke. Performance Analysis And Validation Of A Stereo Vision System. In *IEEE Int. Conf. on Systems, Man and Cybernetics*, volume 2, pages 1409–1416, 2005. DOI 10.1109/ICSMC.2005.1571344.

[61] M. Pollefeys. *Visual 3-D Modeling From Images*. Tutorial notes, http://www.cs.unc.edu/~marc/tutorial.pdf. Accessed September 22nd 2014.

[62] V. Kolmogorov and R. Zabih. Computing Visual Correspondence With Occlusions Using Graph Cuts. In *IEEE Int. Conf. on Computer Vision*, volume 2, pages 508–515, 2001. DOI 10.1109/ICCV.2001.937668.

[63] J. Woo-Seok and H. Yo-Sung. Efficient Disparity Map Estimation Using Occlusion Handling For Various 3-D Multimedia Applications. In *IEEE Transactions on Consumer Electronics, Issue 4*, volume 57, pages 1937–1943, 2011. DOI 10.1109/TCE.2011.6131174.

[64] D. Lowe. Object Recognition From Local Scale-invariant Features. In *The Proceedings of the Seventh IEEE Int. Conf. on Computer Vision*, volume 2, pages 1150–1157, 1999. DOI 10.1109/ICCV.1999.790410.

[65] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded Up Robust Features. In *9th European Conf. on Computer Vision*, pages 404–417, 2006. DOI 10.1007/1174402332.

[66] N. Dalal and B. Triggs. Histograms Of Oriented Gradients For Human Detection. In *IEEE Comp. Society Conf. On Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005. DOI 10.1109/CVPR.2005.177.

[67] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide-baseline Stereo From Maximally Stable Extremal Regions. In *British Machine Vision Computing 2002, Issue 10*, volume 22, pages 761–767, 2004. DOI 10.1016/j.imavis.2004.02.006.

[68] T. Karlsson. Opencv -kirjaston kohteentunnistusmenetelmät, 2009. B.Sc. (Tech) thesis. Elektroniikan, tietoliikenteen ja automaation tiedekunta, Teknillinen korkeakoulu.

[69] C. Harris and M. Stephens. A Combined Corner And Edge Detector. In *Alvey Vision Conference*, volume 15, pages 147–152, 1998.

[70] J. Shi and C. Tomasi. Good Features To Track. In *IEEE Comp. Society Conf. on Computer Vision and Pattern Recognition*, pages 593–600, 1994. DOI 10.1109/CVPR.1994.323794.

[71] H. Hirschmuller. Stereo Processing By Semiglobal Matching And Mutual Information. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Issue 2*, volume 30, pages 328–341, 2008. DOI 10.1109/TPAMI.2007.1166.

[72] C. Zitnick and T. Kanade. A Cooperative Algorithm for Stereo Matching and Occlusion Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000. DOI 10.1109/34.865184.

[73] docs.opencv.org. OpenCV Documentation, stereoRectify function. http://docs.opencv.org/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html. Accessed August 22nd, 2014.

[74] O. Karhu. Sensor System For Environment Perception. Technical report, 2013. Confidential deliverable, FAMOUS project, GIM research group.

[75] IDS Imaging Development Systems GmbH. IDS GigE uEye UI-5120 RE Cameras. http://en.ids-imaging.com/store/produkte/kameras/ui-5120re.html Accessed July 1st 2013.

[76] Camera Calibration Toolbox for MATLAB Tutorial, 2000. http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html. Accessed August 12th 2014.

[77] pointclouds.org Tutorials. List of tutorials for PCL computation, http://pointclouds.org/documentation/tutorials/. Accessed 8th of June 2014.

[78] pointclouds.org Tutorials. Tutorial on PCL VoxelGrid filter, http://pointclouds.org/documentation/tutorials/voxel_grid.php. Accessed 8th of June, 2014.

[79] Radu Bogdan Rusu. *Semantic 3-D Object Maps for Everyday Manipulation in Human Living Environments*. PhD thesis, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.

[80] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm For Model Fitting With Application To Image Analysis And Automated Cartography. *Communications of the ACM*, 24(6):381–395, 1981. DOI 10.1145/358669.358692.

[81] pointclouds.org Documentation. Plane model segmentation tutorial, http://pointclouds.org/documentation/tutorials/planar_segmentation.php#planar-segmentation. Accessed 9th of June 2014.

[82] P. Torr and A. Zissermann. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000. DOI 10.1006/cviu.1999.0832.

[83] Huber P. Robust Statistics. *International Encyclopedia of Statistical Science*, pages 1248–1251, 1981. DOI 10.1007/978-3-642-04898-2-594.

[84] pointclouds.org Documentation. Cylinder model segmentation tutorial, http://pointclouds.org/documentation/tutorials/cylinder_segmentation.php#cylinder-segmentation. Accessed 9th of June, 2014.

[85] P. Forsman. *Three-dimensional Localization And Mapping Of Static Environments By Means Of Mobile Perception*. PhD thesis, Dept. of Automation and Systems Technology, Helsinki University of Technology, 2001. Series A, Research reports 23.

[86] J. Klosowski, M. Held, J. Mitchell, H. Sowizral, and K. Zikan. Efficient Collision Detection using Bounding Volume Hierarchies of k-DOPs. In *IEEE Transactions on Visualization and Computer Graphics*, volume 4, pages 21–36, 1998. DOI 10.1109/2945.675649.

[87] J. Gunther, S. Popov, H. Seidel, and P. Slusallek. Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In *IEEE Symposium in Interactive Ray Tracing*, pages 113–118, 2007. DOI 10.1109/RT.2007.4342598.

[88] C. Ericson. *Real-Time Collision Detection*. 2005. Elsevier Amsterdam/Boston, ISBN 978-1-55860-732-3.

[89] J. Feng, R. Zhong, Y. Yang, and W. Zhao. Quality Evaluation of Spatial Point Cloud Data Collected by Vehicle-borne Laser Scanner. *Int. Workshop on Geoscience and Remote Sensing*, 2008. DOI 10.1109/ETTandGRS.2008.97.

[90] J. Kantz. Application of Sweeping Techniques to Reverse Engineering, 2003. Master's thesis. Dept. of Computer and Information Science, University of Michigan-Dearborn.

[91] A. Fidera. Quality Control Methodologies For 3-D Geometric Primitives Derived From LIDAR Data. 2004.

[92] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[93] G. Nebehay. *Robust Object Tracking Based On Tracking-learning-detection*. 2012. M.Sc. thesis, TU Wien.

Appendix A

Dataset Visualisations

This appendix contains visualisations of the results obtained from the offline datasets and computed using the designed load object measurement software. The subfigures include the 3-D AABB bounding volume length of each axis, the computation times per point cloud per measurement, and minimum and maximum point coordinates in a time series. Dataset A1 (Figure A.1) and A6 (Figure A.4) visualisations are introduced in Section 6.5 in the discussion. Dataset A2 is being discussed in Section 6.4.1. Dataset A3(Figure A.3) is introduced in the introduction of Section 6.5. Datasets B3(Figure A.5) and B5(Figure A.6)are being discussed in Section 7.1. Datasets C3(Figure A.7), C6(Figure A.8), C7(Figure A.9), and D4(Figure A.10) are discussed in Section 6.4.2.

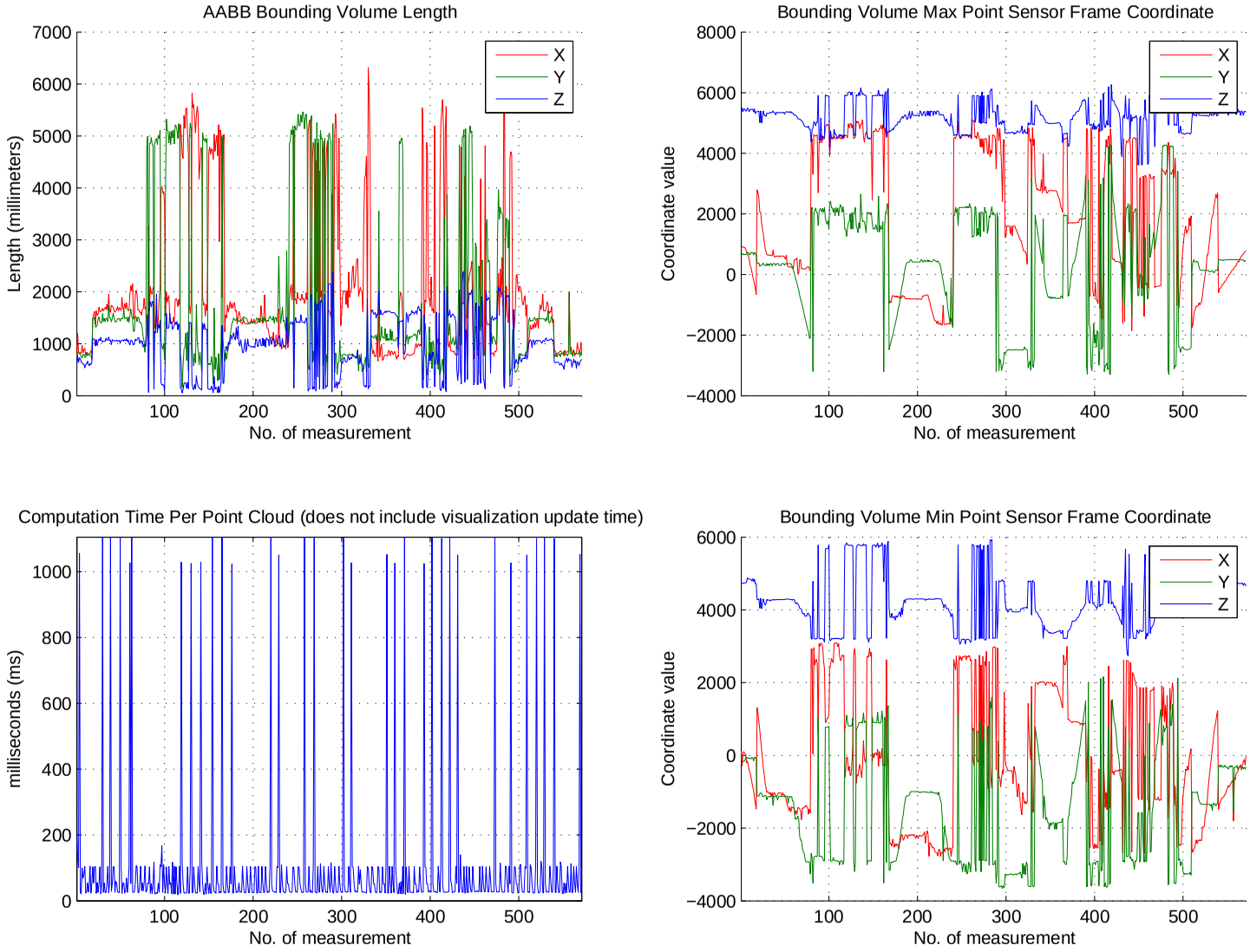

Figure A.1: Bounding volume computation results for dataset A1.

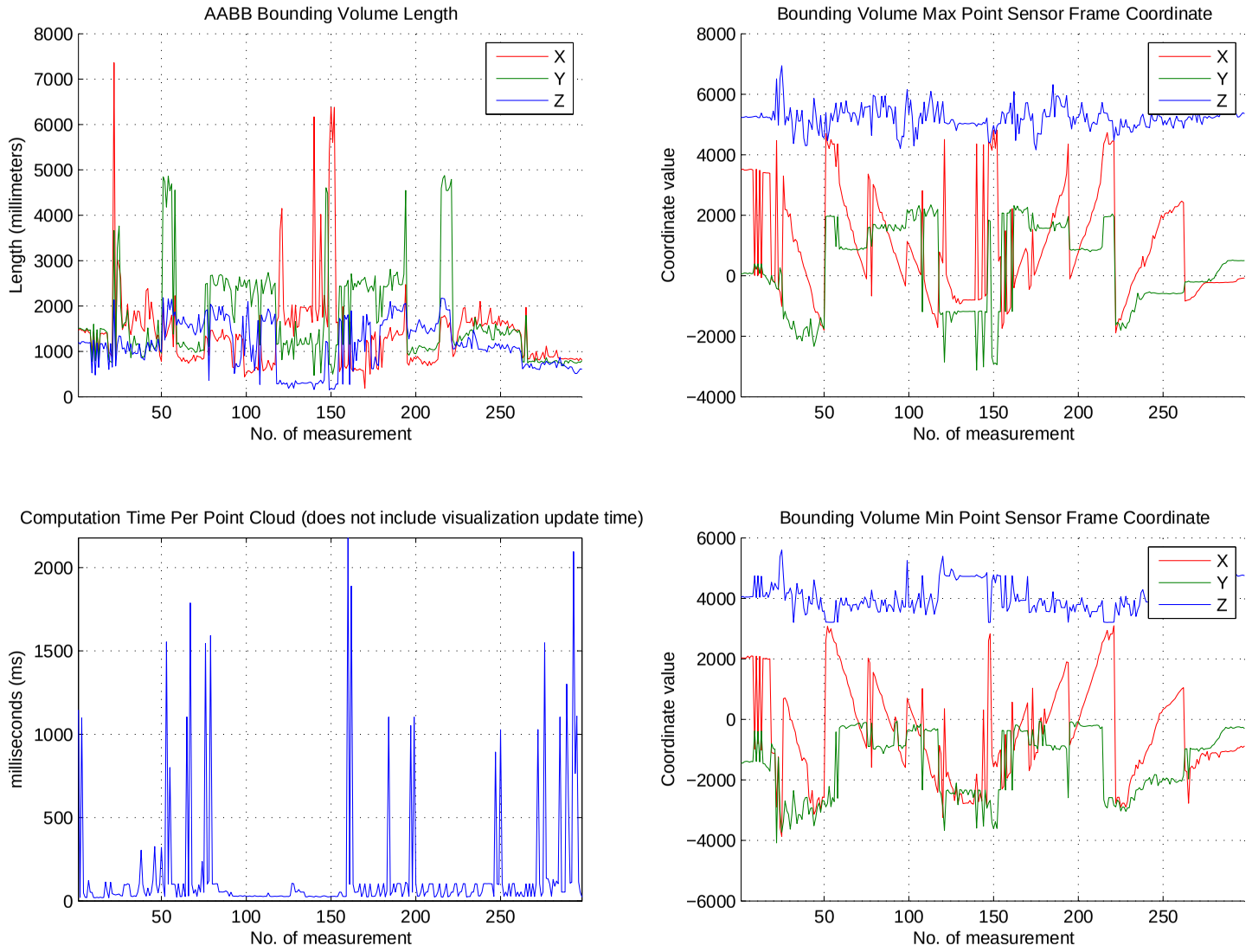

Figure A.2: Bounding volume computation results for dataset A2.

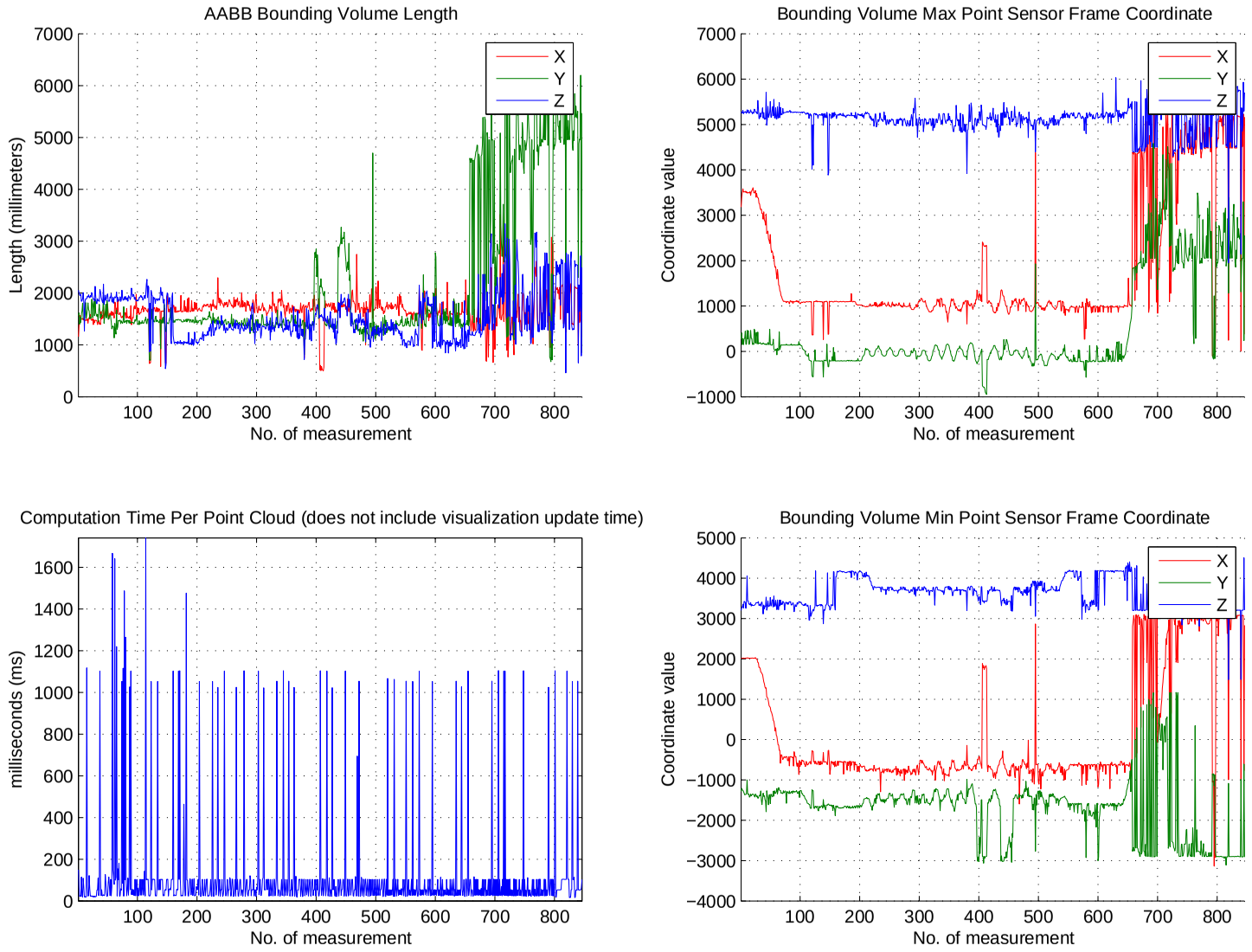

Figure A.3: Bounding volume computation results for dataset A3.

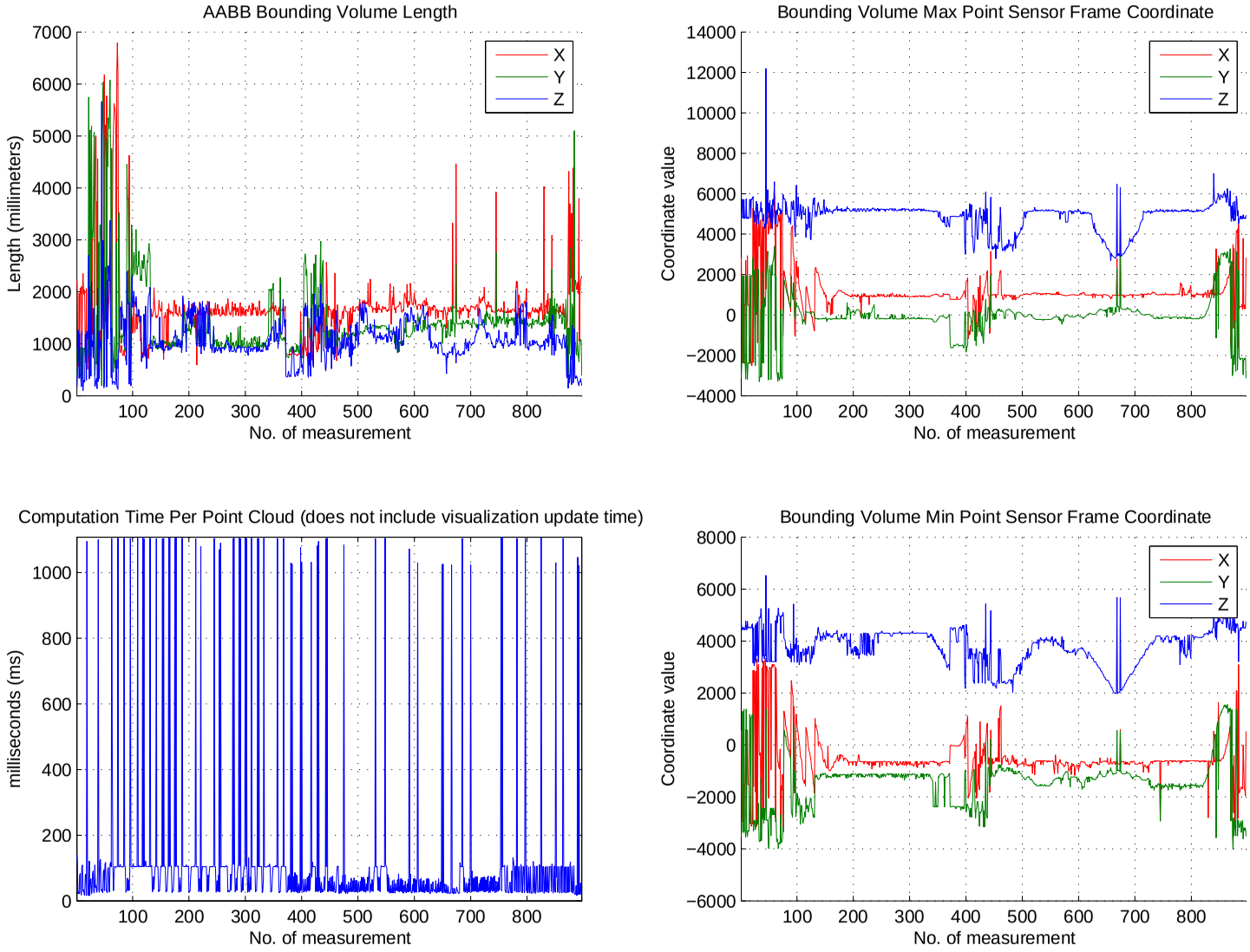

Figure A.4: Bounding volume computation results for dataset A6.

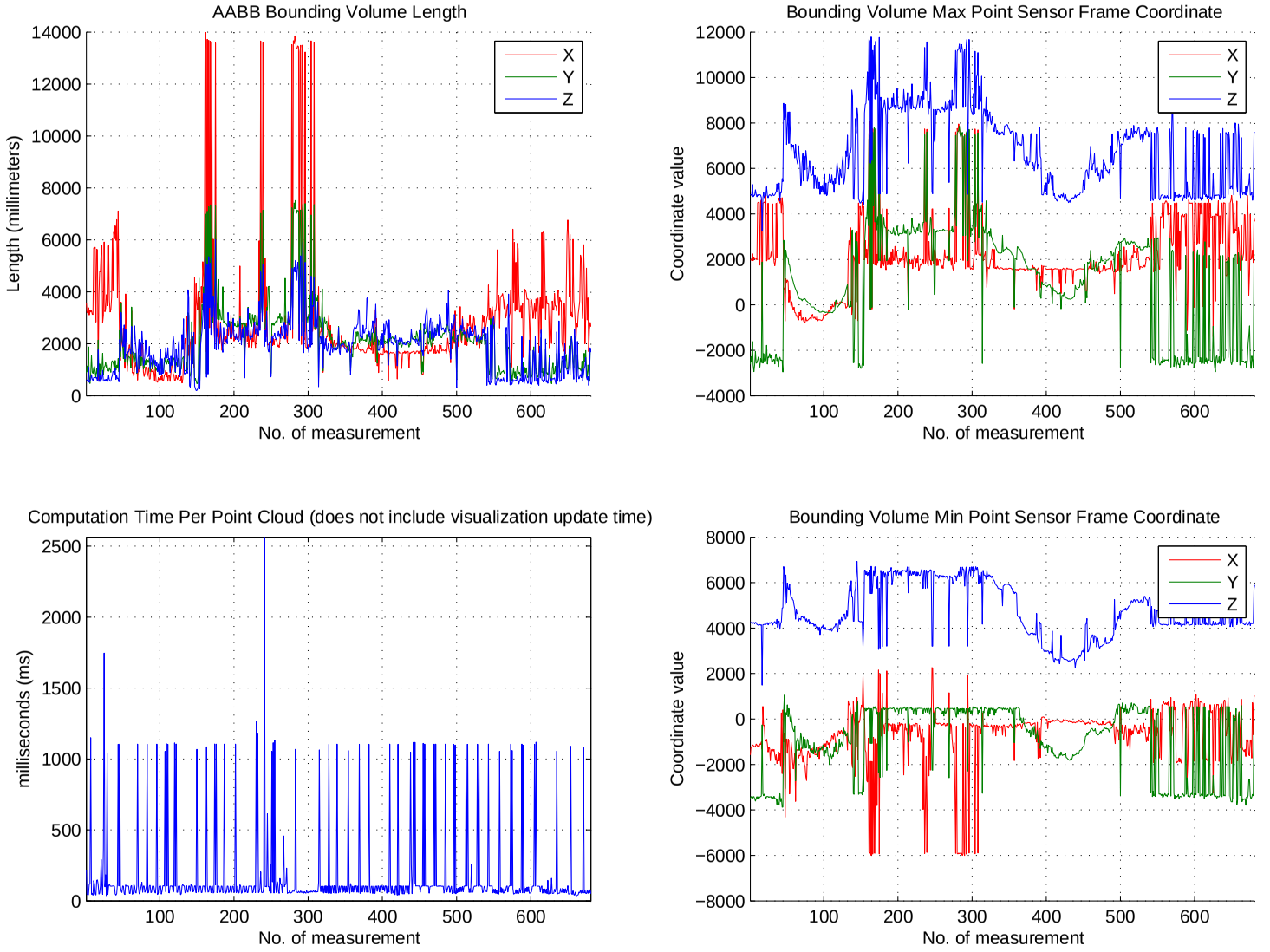

Figure A.5: Bounding volume computation results for dataset B3.

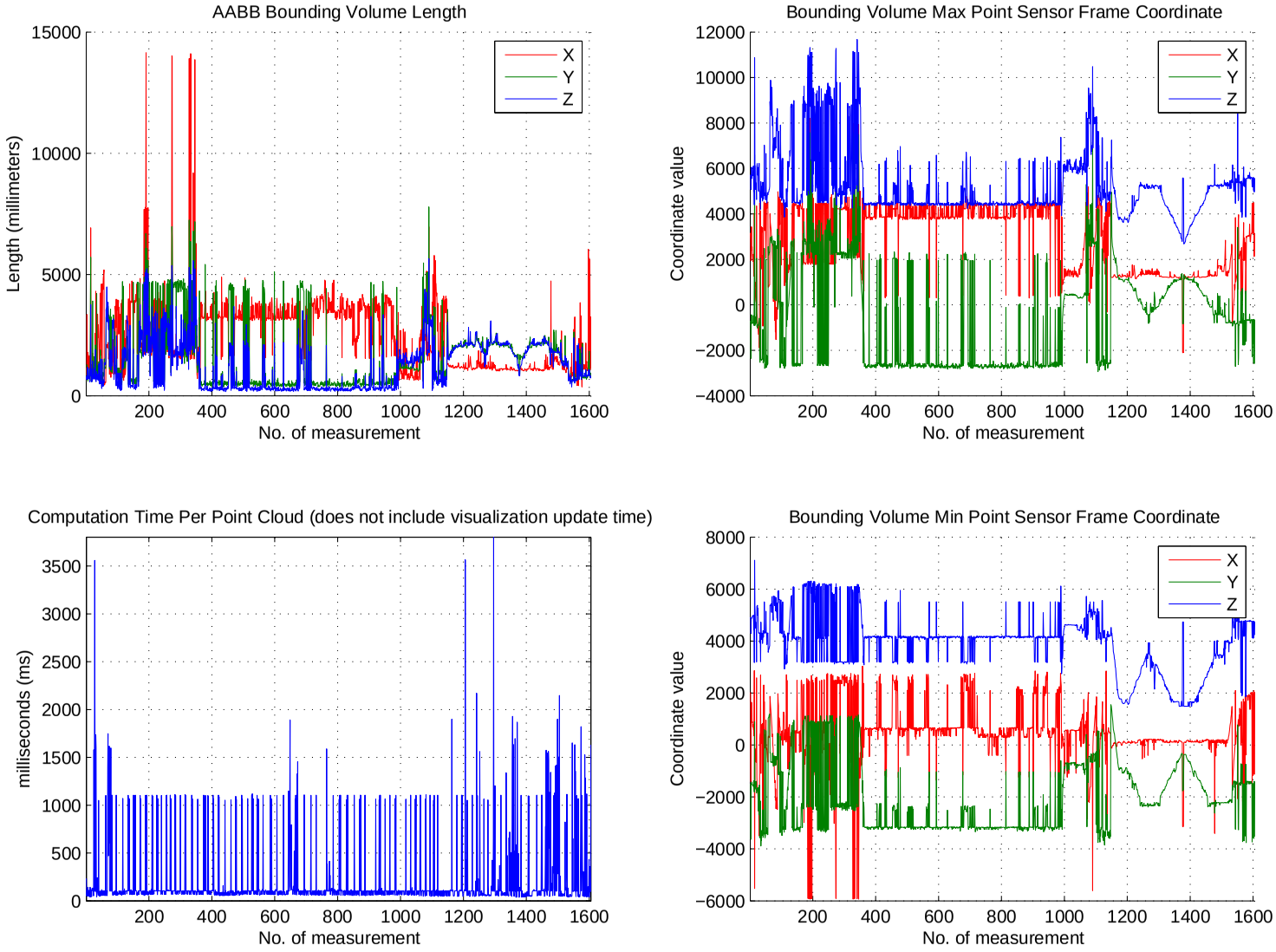

Figure A.6: Bounding volume computation results for dataset B5.

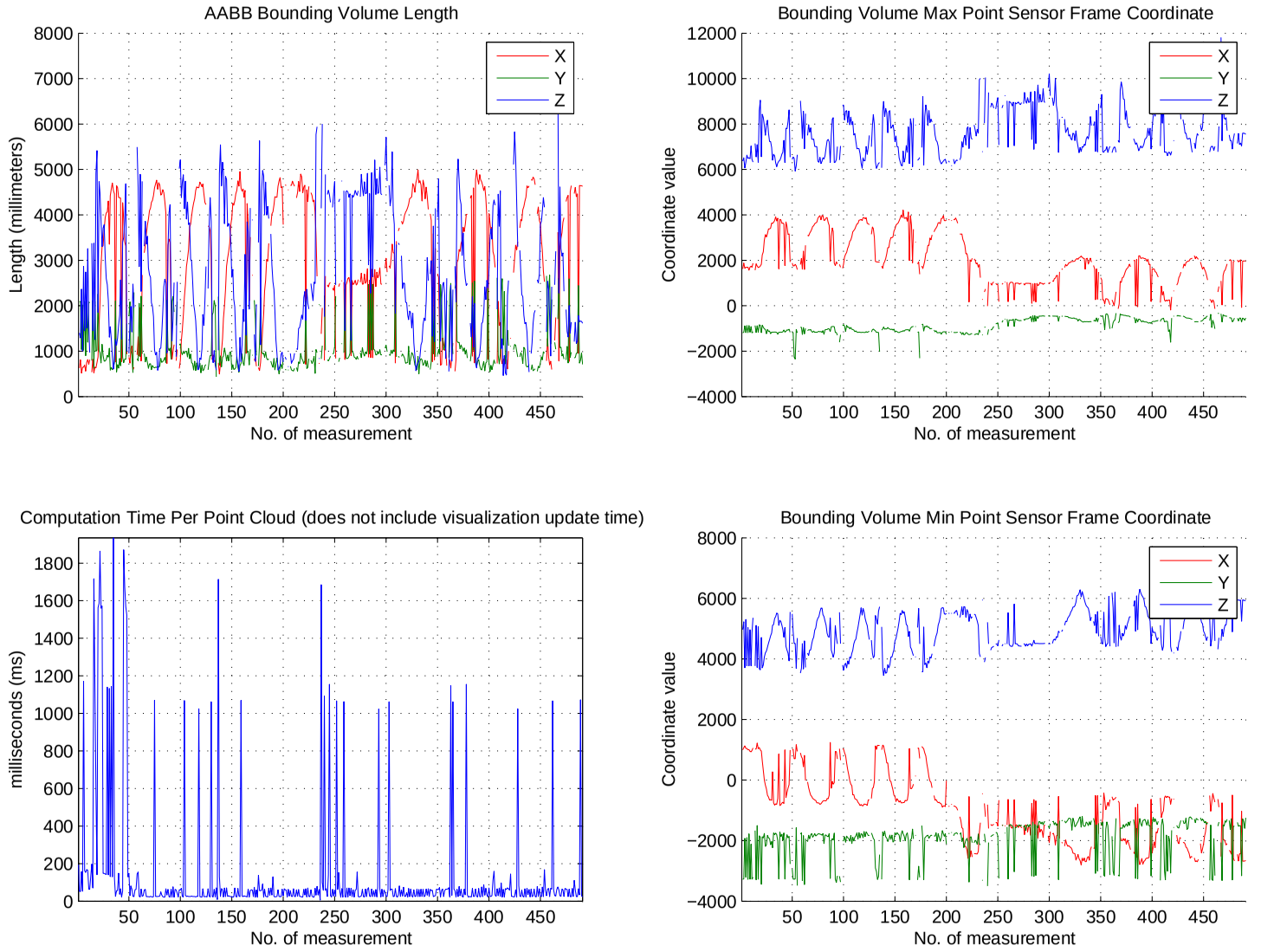

Figure A.7: Bounding volume computation results for dataset C3.

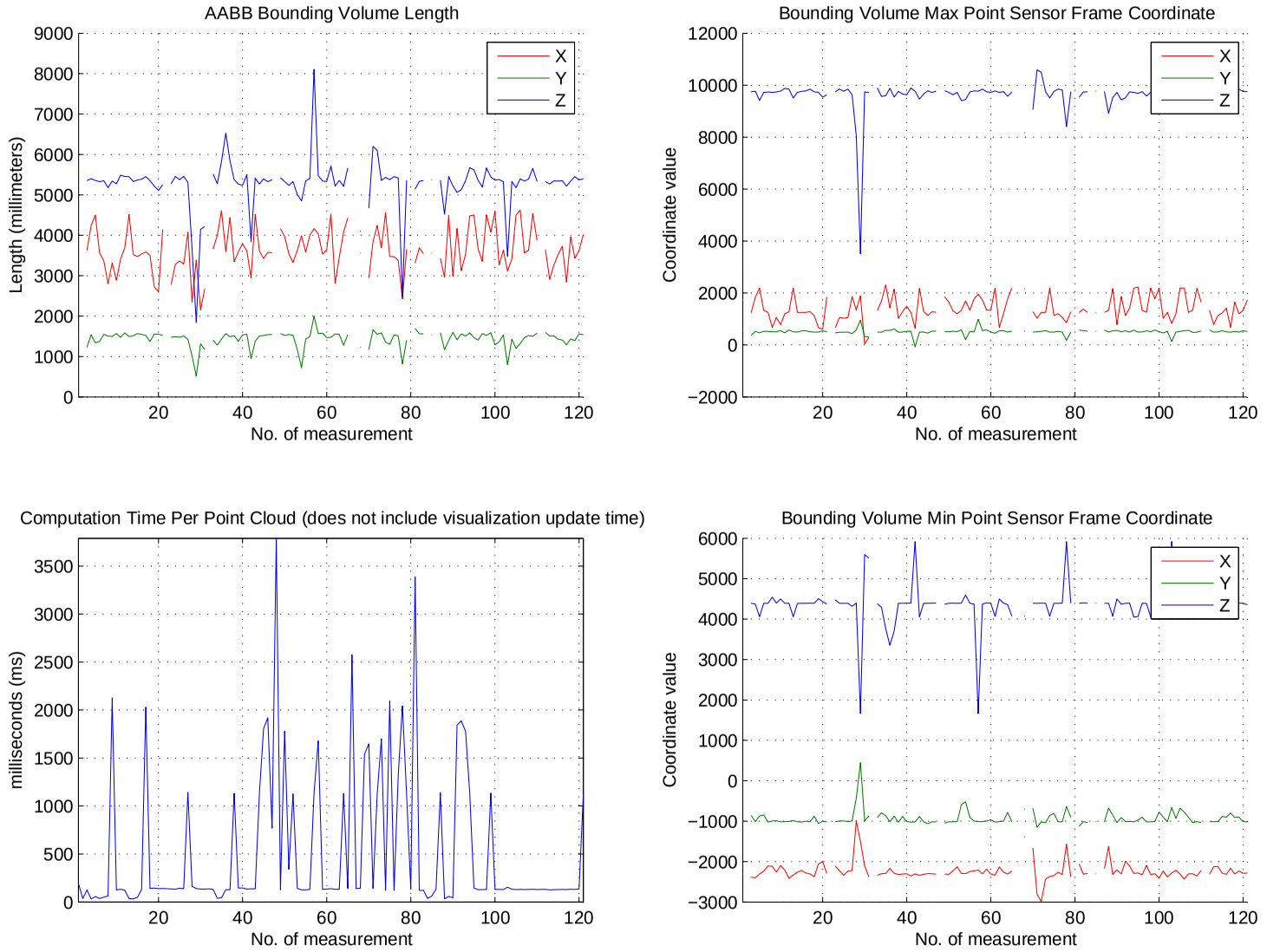

Figure A.8: Bounding volume computation results for dataset C6.

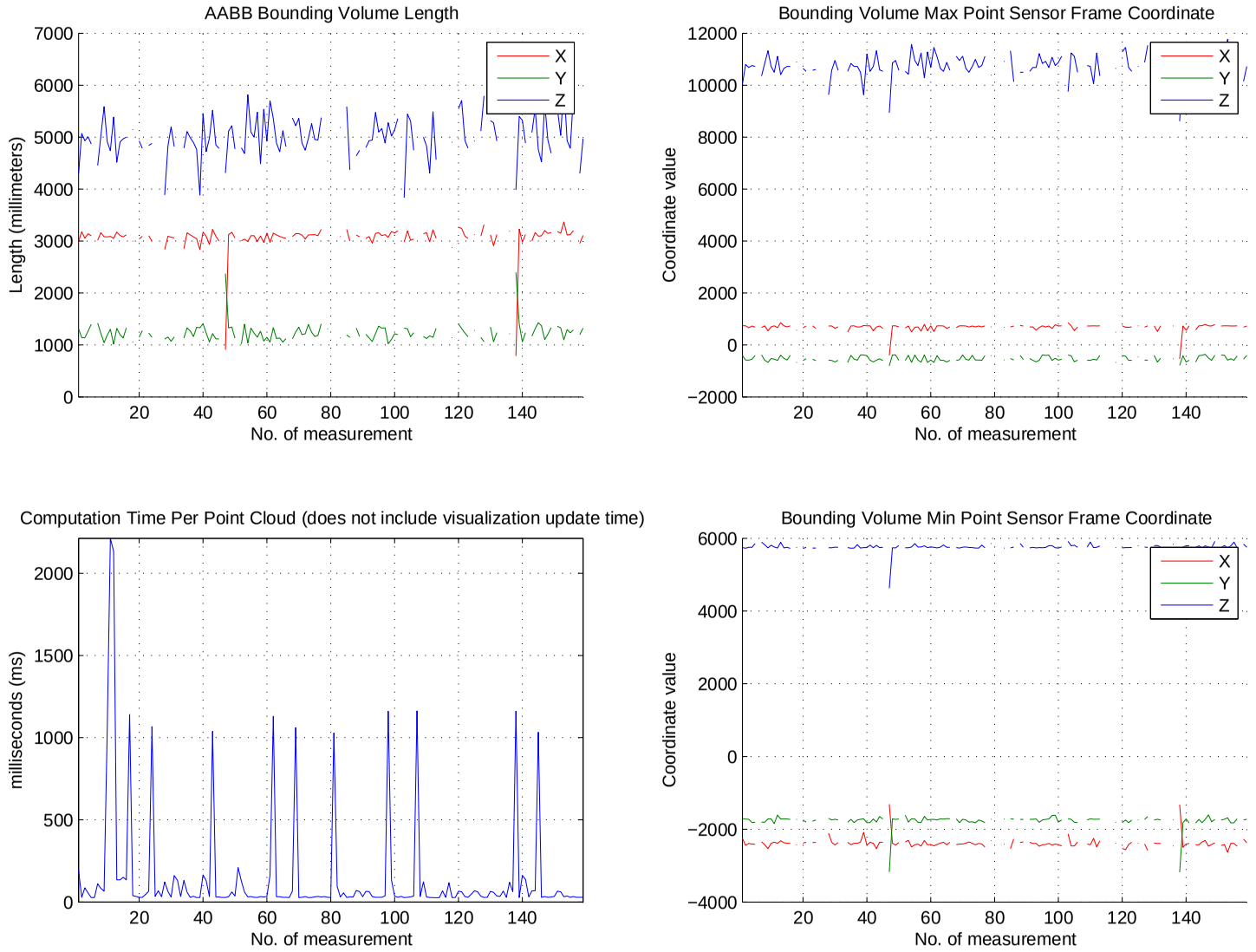

Figure A.9: Bounding volume computation results for dataset C7.

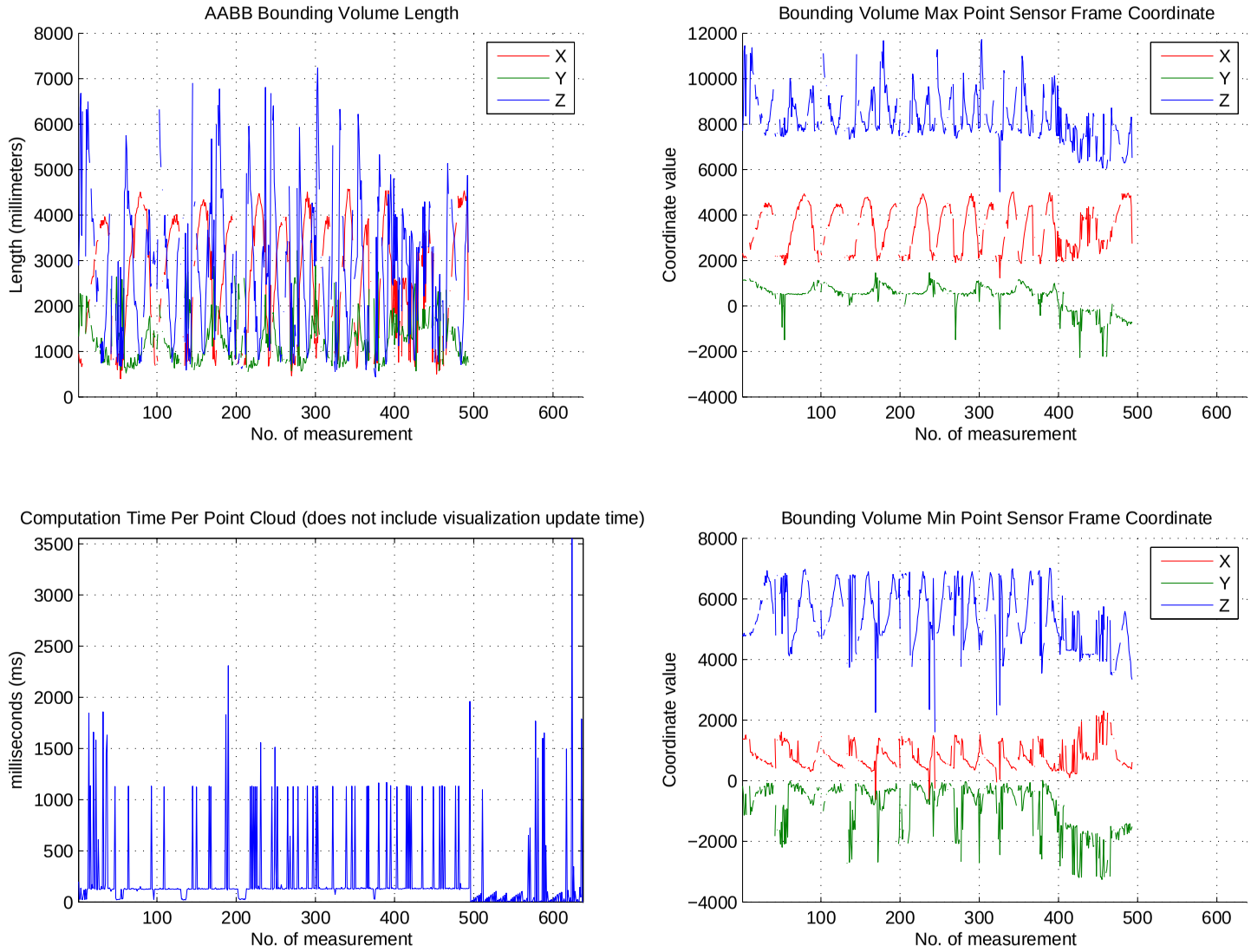

Figure A.10: Bounding volume computation results for dataset D4.

Appendix B

ROS Graphs

This appendix contains ROS component designs that the measurement client software uses. Figure B.1 introduces a ROS network configuration that was used for running the load object measurement client software, the point cloud data publisher, and the 2-D image tracking software. Figure B.1 is referred to in Section 4.2.

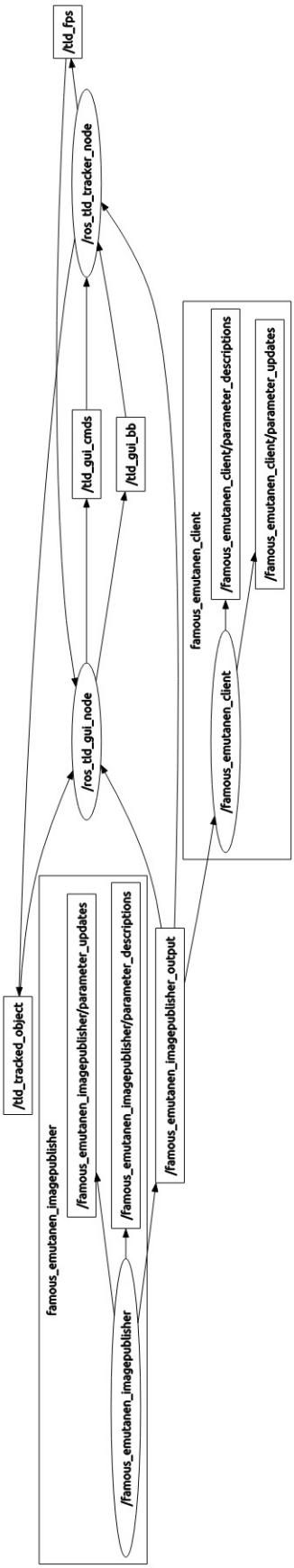

Figure B.1: ROS network used in running offline tests. Visualisation by rqt_graph ROS package.